

THE MINISTRY OF EDUCATION AND SCIENCE OF THE REPUBLIC OF  
KAZAKHSTAN

Kazakh National Research Technical University named after K. I. Satpayev

Institute of Information and Telecommunication Technologies

Department of Cybersecurity, Data Storage and Processing

Keten Timur

Web applications constructor

**DIPLOMA WORK**

specialty 5B070300 – «Information systems»

Almaty 2019

THE MINISTRY OF EDUCATION AND SCIENCE OF THE REPUBLIC OF  
KAZAKHSTAN

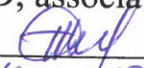
Kazakh National Research Technical University named after K. I. Satpayev

Institute of Information and Telecommunication Technologies

Department of Cybersecurity, Data Storage and Processing

**ADMITTED TO DEFENSE**

Head of Department Cybersecurity  
Data Storage and Processing,  
PhD, associate professor

 N.A. Seilova  
« 13 » 05 2019.


**DIPLOMA WORK**

Theme: «Web applications constructor»


specialty 5B070300 – Information systems

Performed

Keten Timur

Reviewer,  
PhD, professor of AlmaU  
 T.I. Bakibayev  
« 13 » 05 2019.



Scientific advisor,  
Tutor  
 E. A. Ryskylbek  
« 08 » 05 2019.

Almaty 2019

THE MINISTRY OF EDUCATION AND SCIENCE OF THE REPUBLIC OF  
KAZAKHSTAN

Kazakh national research technical University named after K. I. Satpayev

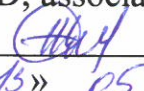
Institute of Information and Telecommunication Technologies

Department of Cybersecurity, Data Storage and Processing

5B070300 – Information system

**AFFIRM**

Head of Department Cybersecurity  
Data Storage and Processing,  
PhD, associate professor

 N.A. Seilova  
« 13 » 05 2019.

**TASK**  
**to perform the Diploma work**

Student Keten Timur

Theme: Web applications constructor

Approved by the order of the University № 1162 from « 16 » 10 2019

Deadline for completion of work « 13 » 05 2019

Source data to diploma work: Web application constructor, web forms , web forms creator, the results of pre-diploma practical work and the literature review, based on theoretical data.

The list of subject to the development of the thesis or a summary of its content:

a) the development of the single page application;

b) the development web form creator;

c) the creation of effective web forms builder.

The list of graphic material presented 15 slides presentation work

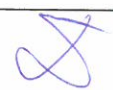
Recommended main literature: 15 sources

**SCHEDULE**  
of preparation of the Diploma work

| The name of the sections, a list of issues         | Deadline for submission to supervisor and consultants | Notice |
|--|---|--------|
| Overview and analysis of existing IS on the market | 10.01.2019 - 08.03.2019.                              |        |
| Writing the functional structure of IS             | 05.02.2019-10.03.2019.                                |        |
| Writing program part                               | 11.03.2019-28.04.2019.                                |        |

**Signature**

Consultants and normocontrol to complete a thesis indicating the related sections of the Diploma work

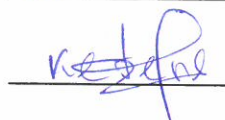
| Section titles   | Consultants,<br>Full name<br>(academic degree, rank)    | Data of signing | Signature  |
|------------------|---|-----------------|--|
| Program software | M. B. Bauyrzhan,<br>Master of Technical Sciences, tutor | 8.05.19         |  |
| Normocontrol     | O.V. Kisseleva,<br>PhD, senior-lector                   | 8.05.19         | O. Kisseleva   |

Scientific adviser



Ryskylbek E. A.

Task was accepted for execution by the student



Keten T.

Date

« 08 » 05 2019.



## REVIEW

**of Diploma work**  
(name of the type of work)

**Keten Timur**  
(student full name)

**Specialty 5B070300 – «Information systems»**

Theme: **Web applications constructor**

Performed:

- a) software part on 11 pages  
b) the volume of work 44 pages

This diploma work is focused developing web applications constructor. The goal of this diploma thesis has been rather ambitious and covered Information System. In introduction chapter author gave overview to basic web form and constructor in the enterprice-level for software solutions.

The second chapter has shown programming languages and software platforms to developing form builder. The last chapter introduces software solution of this taks. For processing author used programming language JavaScript, ReactJS with integrated development environment Visual Studio.

On my mind the work belong to more difficult work, needs to complete knowledge of student. The given taks was completely fulfilled; methods of solution were correctly chosen. Student demonstrated his abilities to process given task including to learn new needed knowledge. Formal level of these diploma work is better than usual; language level I considered very good, so I can to review.

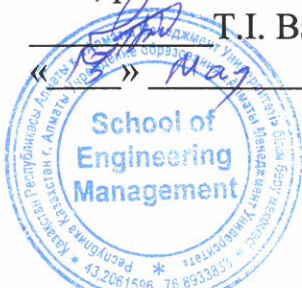
### Performance evaluation

The diploma work is executed on «excellent» (93 %) and its author deserves to have an academic degree «bachelor» in specialty 5B070300 – «Information systems».

### Reviewer

PhD, professor of AlmaU

T.I. Bakibayev  
2019.



**REVIEW**  
**of the scientific advisor**

**DIPLOMA WORK**  
*(name of the type work)*

**Keten Timur**  
*(student full name)*

**5B070300 – Information systems**  
*(profession name and code)*

**Theme: Web applications constructor**

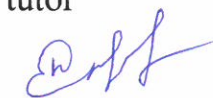
Graduate Keten Timur performed the diploma work according to the schedule and showed himself as a motivated student, able to search self-contained materials and literature in the preparation of the work.

This diploma work is dedicated to developing web applications constructor.

In introduction chapter author gave overview to basic web form and constructor in the enterprise-level for software solutions.

The second chapter has shown programming languages and software platforms to developing form builder. The last chapter introduces software solution of this task. For processing author used programming language JavaScript, ReactJS with integrated development environment Visual Studio.

**Scientific advisor**  
tutor



*signature*

«8» may 2019 y.

**Ryskylbek E.A.**  
*full name*

## Протокол анализа Отчета подобия

заведующего кафедрой / начальника структурного подразделения

Заведующий кафедрой / начальник структурного подразделения заявляет, что ознакомился(-ась) с Полным отчетом подобия, который был сгенерирован Системой выявления и предотвращения плагиата в отношении работы:

**Автор:** Ketan Timur

**Название:** Web application constructor

**Координатор:** Ерсултан Раскулбек

**Коэффициент подобия 1:3,1**

**Коэффициент подобия 2:0,3**

**Тревога:0**

**После анализа отчета подобия заведующий кафедрой / начальник структурного подразделения констатирует следующее:**

- ☒ обнаруженные в работе заимствования являются добросовестными и не обладают признаками плагиата. В связи с чем, работа признается самостоятельной и допускается к защите;
- ☐ обнаруженные в работе заимствования не обладают признаками плагиата, но их чрезмерное количество вызывает сомнения в отношении ценности работы по существу и отсутствием самостоятельности ее автора. В связи с чем, работа должна быть вновь отредактирована с целью ограничения заимствований;
- ☐ обнаруженные в работе заимствования являются недобросовестными и обладают признаками плагиата, или в ней содержатся преднамеренные искажения текста, указывающие на попытки сокрытия недобросовестных заимствований. В связи с чем, работа не допускается к защите.

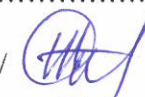
Обоснование:

.....  
.....  
.....  
.....  
.....

.....  
Дата 13.05.192

.....  
Подпись заведующего кафедрой /

начальника структурного подразделения

  
КБДХУ

Окончательное решение в отношении допуска к защите, включая обоснование:



Допускается к защите

Дата

12.05.13г

Подпись заведующего кафедрой /

начальника структурного подразделения



## Протокол анализа Отчета подобия Научным руководителем

Заявляю, что я ознакомился(-ась) с Полным отчетом подобия, который был сгенерирован Системой выявления и предотвращения плагиата в отношении работы:

**Автор:** Ketan Timur

**Название:** Web application constructor

**Координатор:** Ерсұлтан Расқұлбек

**Коэффициент подобия 1:** 3,1

**Коэффициент подобия 2:** 0,3

**Тревога:** 0

**После анализа Отчета подобия констатирую следующее:**

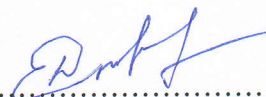
- ☒ обнаруженные в работе заимствования являются добросовестными и не обладают признаками плагиата. В связи с чем, признаю работу самостоятельной и допускаю ее к защите;
- ☐ обнаруженные в работе заимствования не обладают признаками плагиата, но их чрезмерное количество вызывает сомнения в отношении ценности работы по существу и отсутствием самостоятельности ее автора. В связи с чем, работа должна быть вновь отредактирована с целью ограничения заимствований;
- ☐ обнаруженные в работе заимствования являются недобросовестными и обладают признаками плагиата, или в ней содержатся преднамеренные искажения текста, указывающие на попытки сокрытия недобросовестных заимствований. В связи с чем, не допускаю работу к защите.

Обоснование:

.....  
.....  
.....  
.....  
.....

13.08.19

Дата



Подпись Научного руководителя



|  |                             |
|--|-----------------------------|
| Университет:                               | Satbayev University         |
| Название:                                  | Web application constructor |
| Автор:                                     | Keten Timur                 |
| Координатор:                               | Ерсұлтан Расқұлбек          |
| Дата отчета:                               | 2019-05-03 11:10:49         |
| Коэффициент подоби́я № 1:                  | 3,1%                        |
| Коэффициент подоби́я № 2:                  | 0,3%                        |
| Длина фразы для коэффициента подоби́я № 2: | 25                          |
| Количество слов:                           | 11 042                      |
| Число знаков:                              | 67 283                      |
| Адреса пропущенные при проверке:           |                             |
| Количество завершенных проверок:           | 19                          |

#### Самые длинные фрагменты, определенные, как подобные

| № | Название, имя автора или адрес гиперссылки<br>(Название базы данных)  | Автор | Количество<br>одинаковых<br>слов |
|---|---|-------|----------------------------------|
| 1 | URL_<br><a href="https://code.visualstudio.com/Docs">https://code.visualstudio.com/Docs</a>   |       | 38                               |
| 2 | URL_<br><a href="https://f8gure.com/orange-county-web-design/">https://f8gure.com/orange-county-web-design/</a>   |       | 19                               |
| 3 | URL_<br><a href="https://code.visualstudio.com/Docs">https://code.visualstudio.com/Docs</a>   |       | 17                               |
| 4 | URL_<br><a href="https://steemit.com/introduction/@kostiayatsuk/lesson-1-javascript">https://steemit.com/introduction/@kostiayatsuk/lesson-1-javascript</a> |       | 16                               |



|    |   |                                 |    |
|----|---|---------------------------------|----|
| 5  | URL_<br><a href="https://code.visualstudio.com/Docs">https://code.visualstudio.com/Docs</a>   |                                 | 14 |
| 6  | URL_<br><a href="https://steemit.com/introduction/@kostiyatsuk/lesson-1-javascript">https://steemit.com/introduction/@kostiyatsuk/lesson-1-javascript</a>   |                                 | 13 |
| 7  | URL_<br><a href="https://code.visualstudio.com/Docs">https://code.visualstudio.com/Docs</a>   |                                 | 12 |
| 8  | URL_<br><a href="https://www.enhancedtech.com/blog/upgrade-to-avoid-end-of-support-for-sql-server-2008/">https://www.enhancedtech.com/blog/upgrade-to-avoid-end-of-support-for-sql-server-2008/</a> |                                 | 12 |
| 9  | Porównanie narzędzi do tworzenia aplikacji typu SPA na przykładzie Angular2 i React<br><i>Politechnika Lubelska (Politechnika Lubelska)</i>   | Jadwiga<br>Dorota<br>Kalinowska | 12 |
| 10 | URL_<br><a href="https://countwordsfree.com/js-formatter">https://countwordsfree.com/js-formatter</a>   |                                 | 10 |

#### Документы, в которых найдено подобные фрагменты: из RefBooks

Не обнаружено каких-либо заимствований

#### Документы, содержащие подобные фрагменты: Из домашней базы данных

Не обнаружено каких-либо заимствований

#### Документы, содержащие подобные фрагменты: Из внешних баз данных

Документы, выделенные жирным шрифтом, содержат фрагменты потенциального плагиата, то есть превышающие лимит в длине коэффициента подобия № 2

| № | Название<br>(Название базы данных)  | Автор                        | Количество<br>одинаковых<br>слов<br>(количество<br>фрагментов) |
|---|---|------------------------------|--|
| 1 | Porównanie narzędzi do tworzenia aplikacji typu SPA na przykładzie Angular2 i React<br><i>Politechnika Lubelska (Politechnika Lubelska)</i> | Jadwiga Dorota<br>Kalinowska | 21 (2)   |
| 2 | Przeglądarkowa symulacja wybranych układów dynamicznych<br><i>Politechnika Śląska (RAu - WYDZIAŁ AUTOMATYKI, ELEKTRONIKI i )</i>            | Jan Śladek                   | 10 (1)   |

#### Документы, содержащие подобные фрагменты: Из интернета

Документы, выделенные жирным шрифтом, содержат фрагменты потенциального плагиата, то есть превышающие лимит в длине коэффициента подобия № 2

| № | Источник гиперссылки  | Количество<br>одинаковых<br>слов<br>(количество<br>фрагментов) |
|---|---|--|
| 1 | URL_<br><a href="https://code.visualstudio.com/Docs">https://code.visualstudio.com/Docs</a> | 93 (6)   |



|   |   |         |
|---|---|---------|
| 2 | URL_<br><a href="https://www.microsoft.com/en-us/sql-server/sql-server-2017-features">https://www.microsoft.com/en-us/sql-server/sql-server-2017-features</a>                                       | 62 (10) |
| 3 | URL_<br><a href="https://f8gure.com/orange-county-web-design/">https://f8gure.com/orange-county-web-design/</a>   | 49 (6)  |
| 4 | URL_<br><a href="https://steemit.com/introduction/@kostiayatsuk/lesson-1-javascript">https://steemit.com/introduction/@kostiayatsuk/lesson-1-javascript</a>   | 48 (5)  |
| 5 | URL_<br><a href="https://countwordsfree.com/js-formatter">https://countwordsfree.com/js-formatter</a>   | 25 (3)  |
| 6 | URL_<br><a href="https://en.wikipedia.org/wiki/Visual_Studio_Code">https://en.wikipedia.org/wiki/Visual_Studio_Code</a>   | 15 (2)  |
| 7 | URL_<br><a href="https://www.enhancedtech.com/blog/upgrade-to-avoid-end-of-support-for-sql-server-2008/">https://www.enhancedtech.com/blog/upgrade-to-avoid-end-of-support-for-sql-server-2008/</a> | 12 (1)  |
| 8 | URL_<br><a href="https://en.wikipedia.org/wiki/Big_data">https://en.wikipedia.org/wiki/Big_data</a>   | 5 (1)   |

Copyright © Plagiat.pl 2002-2019

## ANNOTATION

This diploma is related to the development of a client-server system, is the creation of an effective and easy-to-use web forms constructor.

As initial materials, web applications of companies and websites were taken for creating a form for a website of any complexity without much effort. In the process of creating a web site such programming languages as JavaScript (ECMAScript 6) were used. I use React and Redux for the front-end, Node.js as the platform, Express as the web framework, and database.

The client-server system processes and verifies information about users. A full responsive form builder that interacts with the json endpoint to load and save the generated forms.

## АННОТАЦИЯ

Данная дипломная работа связана с разработкой клиент-серверной системы которая является создание эффективного и простого для использования конструктора веб-форм.

В качестве исходных материалов были взяты веб приложения компаний и сайты предназначенные для созданий формы для сайта любой сложности без особых усилий. В процессе создания веб сайта были использованы такие языки программирования как JavaScript (ECMAScript 6). Также React и Redux для внешнего интерфейса, Node.js в качестве платформы, веб-фреймворк Express и базы данных.

Клиент-серверная система обрабатывает и проверяет информацию о пользователях. Польностью гибкий конструктор форм, который взаимодействует с конечной точкой JSON для загрузки и сохранения сгенерированных форм.

## АНДАТПА

Бұл дипломдық жұмыс тиімді және қарапайым веб-форматтардың конструкторын құру болып табылатын клиент-серверлік жүйені құрумен байланысты.

Бастапқы материалдар ретінде компаниялардың веб-сайттары және веб-қосымшалары алынды. Веб-сайт құру барысында JavaScript (ECMAScript 6) сияқты бағдарламалау тілдері пайдаланылды. Мен React және Redux-ді сыртқы интерфейс үшін, Node.js-ді платформа ретінде, веб-фреймворк ретінде Express және дерекқор пайдаландым.

Клиент-сервер жүйесі пайдаланушылар туралы ақпаратты өңдейді және тексереді. Жасалған пішіндерді жүктеу және сақтау үшін JSON соңғы нүктесімен өзара әрекеттесетін және толық жауап беретін веб-форма конструкторы.

## CONTENT

|     |   |    |
|-----|---|----|
|     | Introduction  | 7  |
| 1   | Web-application Basics  | 8  |
| 1.1 | Basic functions of Web-applications Constructor                       | 9  |
| 1.2 | Fundamentals of Web form  | 10 |
| 1.3 | Setting system tasks:   | 17 |
| 2   | Methods of implementing Information System                            | 18 |
| 2.1 | Single page application. Advantages of single page application.       | 18 |
| 2.2 | Source code editor Visual Studio Code                                 | 20 |
| 2.3 | Node.js software platform   | 22 |
| 2.4 | JavaScript library React.js   | 24 |
| 2.5 | ASP.NET Core 2.1/2.2 framework  | 27 |
| 2.6 | NoSQL database program MongoDB  | 28 |
| 3   | Realization software for Information system                           | 31 |
| 3.1 | Structure of programming realization                                  | 31 |
| 3.2 | Justification of the choice of programming languages                  | 32 |
| 3.3 | Description of the developed application software                     | 33 |
| 3.4 | Realization of Information system. Creating a single page application | 34 |
|     | Conclusion  | 38 |
|     | References  | 39 |
|     | Appendix A  | 40 |
|     | Appendix B  | 41 |
|     | Appendix C  | 44 |

## INTRODUCTION

Constructor is a web-based platform for creating enterprise-level software solutions. Allows creating both simple and fairly complex software solutions right from the web browser window. Good for creating complex and specialized solutions.

Creating forms for a site is sometimes a complicated and time-consuming business. Surely, you really value your time and would not want to spend the whole day doing this. Especially if you do not want to delve into the "technical kitchen" of cooking forms and learn programming languages. But in order to create the simplest form, you need to know at least the HTML markup language and the language for describing the appearance of CSS in order to decorate your form beautifully. Moreover, to make the form more interactive, add some visual effects or hints, you should know the programming language.

For many, all this will turn out to be terrible and difficult not only for application, but also for understanding. What if you just need to make a form and at the same time, as quickly as possible? And do not want to delve into any programming at all? How to be? Of course, you can order the development of the form from the programmer. But there is another option that will help you create your form without the help of a programmer - Web Forms Constructor.

Web forms have revolutionized many areas, including real estate, medicine, finance, commerce, and many other industries where documentary work and documentation play a vital role. Now what was previously done manually on paper is now easily and quickly done online. .

Actuality of this work is that the scope of the use of the designer of web- forms is not limited to certain industries or types of sites. They can be used to solve a variety of tasks in many areas.

The purpose of this work is to create an effective and easy-to-use web-forms constructor. This tool is ideal for use by anyone who does not want to delve into the technical parts of the program and understand programming. He will create a form for the site of any complexity without much effort.

To achieve the purpose, the following tasks are required: a study of the theoretical part about creating web-applications constructor, analyzing existing web forms constructor, using the necessary and effective tools and software; optimization of constructor functions.



## 1 Web-application Basics

A web-application is a solution based on the interaction between a browser and a web server. Such applications are cross-platform services accessible from any modern device and are not tied to the network architecture: you can access them from a local computer or from a smartphone on the other side of the world using a convenient protocol, such as the fastest or encrypted.

A web application is traditionally divided into two parts: client and server. The client part, or simply the client, is the “face” of the application, what the user sees. First of all, it is responsible for the interface and direct user interaction. To perform complex operations, the client forms requests to the server and processes responses from it.

The server part, or server, is the “brain” of the application, where all complex calculations are performed, volumetric data are stored, and work is coordinated as a whole. Millions of client systems can simultaneously interact with one server.

This architecture allows you to divide the areas of responsibility between the two subsystems and make them more independent. Significant advantage of building web applications to support standard browser functions is that functions should be performed independently of the client’s operating system. Instead of writing different versions for Microsoft Windows, Mac OS X, GNU / Linux and other operating systems, the application is created once for a randomly selected platform and is deployed on it. However, different implementations of HTML, CSS, DOM, and other specifications in browsers can cause problems when developing web applications and subsequent support. In addition, the ability of a user to customize many browser settings (for example, font size, colors, disabling script support) may prevent the application from working correctly:

- the web application consists of client and server parts, thereby implementing the client-server technology;
- the client part implements the user interface, forms requests to the server and processes responses from it;
- the server part receives a request from the client, performs calculations, then generates a web page and sends it to the client over the network using the HTTP protocol.

The web application itself can act as a client of other services, for example, a database or another web application located on another server. A striking example of a web application is the content management system for Wikipedia articles: many of its participants can take part in creating a network encyclopedia using browsers of their operating systems (be it Microsoft Windows, GNU / Linux or any other operating system) and without downloading additional executables. modules for working with articles database. Currently gaining popularity is a new approach to developing web applications, called Ajax. When using Ajax, web application pages do not reload entirely, but only load the necessary data from the server, which makes them more interactive and productive [1].

## 1.1 Basic functions of Web-applications Constructor

The application is a tool for developing application solutions in web-technology. Designing screen forms on the web based on the requirements for automating business processes of the bank provides the ability to organize mobile workstations for access by end users in any modern browser on any operating system.

A web application is a client-server application in which a client interacts with a server using a browser, and a web server is responsible for the server. The web application logic is distributed between the server and the client, data is stored primarily on the server, information is exchanged over the network. One of the advantages of this approach is the fact that clients are independent of the user's specific operating system, therefore web applications are cross-platform services.

Main functions:

- support the development of screen forms on the web using various controls;
- ability to customize the properties of controls (visibility, accessibility, styling, etc.);
- ability to organize controls on screen form in the web (order of arrangement, alignment, centering, etc.);
- support for the Eclipse cross-platform IDE development environment, which allows using a large repository of extensions, including;
- development of screen forms on the web in accordance with the principles of the model-view-controller design pattern (MVC);
- support for mechanisms for re-using already created screen forms on web;
- deployment (deploy) of developed web-applications on servers.

Web-Application Constructors are services that allow you to create applications for specific needs (for example, food delivery or taxis) without programming at all, based on predefined templates with the addition of necessary widgets and design options.

Web applications require very little disk space (or computing power) on the client. All the client does is display the data. In many cases, the data is stored remotely too. As with other cloud computing, this can allow easy communication.

Typical Web Application using:

- the use of web applications brings certain benefits both to website visitors and their developers. Web applications allow visitors to quickly and easily find the information they need on websites with a lot of information;
- web applications allow you to collect, save and analyze data received from site visitors. The web application can be used to update websites with periodically changing content. The web application frees the web designer from the routine work of constantly updating the site's HTML pages [2].

Support the development of screen forms on the web using various controls:

- input field, tables;
- drop-down lists, buttons;
- icons, etc.

## 1.2 Fundamentals of Web form

At its core, a web form is a specifically restricted area on a website page. In these areas, the site visitor can enter one or information, as well as select specific actions from the proposed ones.

The web form on the website is an analogue of a paper form, a questionnaire, a form and a questionnaire.

Forms have fields to fill out, as well as lists and radio buttons that allow the user to select one or more items.

The form can be used to receive information from site visitors. For example, using a web form you can collect personal data, information about orders, information that is necessary for invoicing, delivery methods, and so on. Visitors are required to enter various types of information in the form field.

The above information can be set by configuring radio buttons, checkboxes and drop-down lists and entering information into text fields. It is also possible to specify the method of collecting information that is entered by site visitors, as well as to specify the way the results data is displayed on the confirmation page that the user is viewing.

Placing web forms should be carefully thought out depending on their purpose: subscription form, order form, callback form or something else. The user should be comfortable working with the site, and the most important web forms for the company should be given the opportunity to reach without a scroll. For example, it is better to place the web login form in your personal account or callback at the top of the page: it immediately gets into the attention of the visitor - this is how the company's interaction with a potential client begins. Try to place the most important information about your product or service, a call to action and a web form to the site's "fold line" in order to immediately attract the user's attention.

Web forms in a multi-column format often repel site visitors. Studies of eye movement - eye tracking (eye-tracking) show that it is more convenient for the user to "scan" the form from top to bottom than from left to right. Thus, a single-column web form is the best solution for collecting information.

Each web form has its own structure which depends on the amount of information needed by the site owner to provide services, sell goods or register for an event. In this subtle question, it is important to keep the balance of the number of fields to fill. Much depends on the visual component of your form. Be consistent in requesting information, build the logic of the form so that the user understands you.

In the table 1.1 we see that main five components of form creation with descriptions like a structure, label fields and etc.

Help the user fill out the form, add tooltips in the right places, show the percentage of the form, be sure to sign the input fields. Make your form beautiful, add animations, remove template fields and make your own, nice and modern, dilute them with icons - your website visitors will thank you [3].

And the next one table 1.2 two elements of form.

Table 1.1– Form creation consists of five components

| № | Component     | Description  |
|---|---------------|--|
| 1 | Structure     | This includes the order of the fields, their appearance on the page in the form, and the logical connection between the input fields.  |
| 2 | Input fields  | They include text fields, password fields, checkboxes, radio buttons, and any other ways to enter the necessary information.   |
| 3 | Label fields  | Indicate what needs to be entered in the fields.   |
| 4 | Action button | By pressing this button, there is some kind of action - for example, the data is sent to the server.   |
| 5 | Feedback      | The user wants to understand whether he has entered the information correctly - and for this purpose, feedback is used. Most often, this is a simple text message that notifies you of a positive result (“Registration completed!”), Or a negative one (“The number you entered is incorrect”). |

Table 1.2 – Forms can include the following elements

| № | Elements   | Description  |
|---|------------|--|
| 1 | Tips       | Help the user to understand exactly what to enter into the form. |
| 2 | Validation | Help the user to understand exactly what to enter into the form. |

### 1.2.1 Web form structure and elements.

Web form is one of the types of communication. And, like any conversation, it should be a logical connection between the user and the application or site being used. Request only the necessary information.

Make sure you ask the user for really important information. The more input fields - the worse the conversion rate, therefore, always consider why you need this or that information, and what it will be used for.

Structure the form logically. Query information logically from the user's point of view, not the site or database. For example, it would be unusual to first ask the address of the user, and only then his name.

Grouping related information. Group the information connected logically into separate blocks. This will facilitate the understanding of what will need to be entered in a single block, and will speed up the user's “dialogue” with the system. Take a look at how this works, on the example of the contact information form below.

For example in figure 1.1 we see the accessibility tree of form in HTML page. The accessibility tree and the DOM tree are parallel structures. Roughly speaking the accessibility tree is a subset of the DOM tree. It includes the user interface objects of the user agent and the objects of the document.

Every web form must be wrapped in <form> tags. In most cases, all of the form fields will be nested between these tags. There are several attributes that may



optional be used with the form element, including: accept-charset: This optional attribute is used to identify the character encodings acceptable to the server and code processing form input. If more than one encoding is specified, one space should be placed between each encoding. If left blank or not provided, the encoding will default to the same encoding as the document containing the form.

The form elements into which you can enter text are called form fields. The form fields may already contain their name (Review, Reset, Submit, etc.) or allow you to enter text. The form fields are essentially text fields, radio buttons, buttons, and other elements. The choice of certain elements is influenced by the information that must be obtained from the visitor.

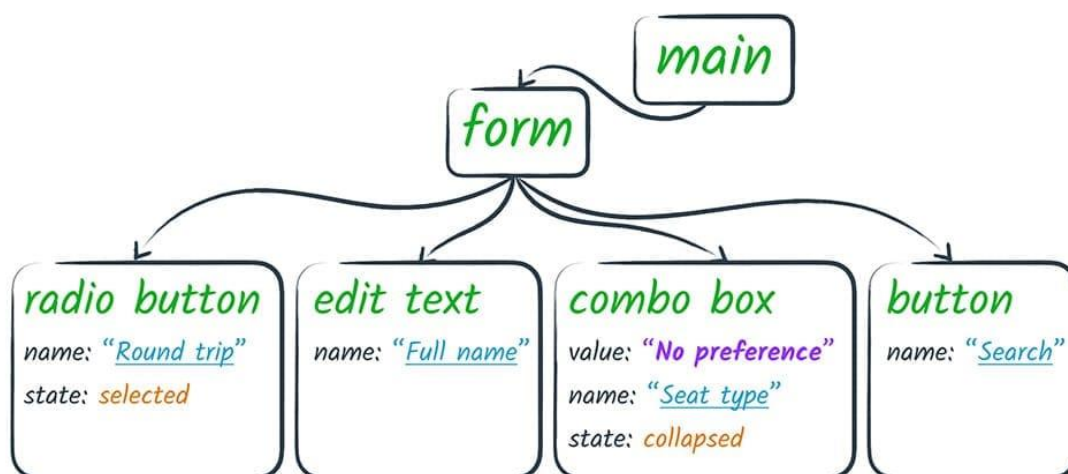


Figure 1.1 – The Accessibility tree

The text box is used to enter text strings. The maximum number of characters that can be entered in this field does not depend on its width.

Checkboxes are used to select additional items or services. The visitor can independently uncheck the box or set it. With the help of checkboxes you can set the ability to select multiple items at once. It should be noted that the flags have only two values: “set” or “reset”.

As for radio switches, they are used instead of checkboxes in the situation when a site user has to choose one of several values. One of these switches, as a rule, is valid by default. Selecting the second switch resets the previous value.

The text area (textarea) is used to enter one or more lines of text. This field scrolls, which makes it possible to enter texts of various sizes into it.

Text areas have wide application possibilities (for example, using them you can create guestbook).

A drop-down list (select) is used in case you need to present a list of different options to the user. The functions of these lists are similar to the functions of switches, but lists do not take up much space in forms. The drop-down list can be configured to allow the selection of one value or several. Very often such a list is used as a drop-down menu in the navigation of an Internet resource.

The button is used to send the completed form when clearing fields or performing other actions. The visitor only needs to click on one or another button.

In addition, the form may contain a picture that can be used as a button. After the visitor fills out the form, he needs to click on this picture, and all the information from the form will be sent to the script that processes the forms:

- using the button, you can change these or other settings. It is possible to use fonts, colors and tables;
- using the group window, you can isolate a group of elements or text from other information that is available on the page.

Site visitors may have the ability to send any files to the site. If the form has a field called “file transfer”, the user can click on the button called “Browse”, select the desired file and send it.

After you select the type of fields that are added to the form, you can determine their appearance and purpose. The name of the field and instructions for their use can be entered into the form. For each field properties can be set. For example, you can set the length of the text field, specify the default switch, and define values in the drop-down list.

Next, you need to set validation rules. For example: you can specify required fields in the form or specify that you only need to enter data of a certain type into certain fields (for example, only letters or only numbers; enter email addresses; Entering url addresses). Validation rules ensure that a visitor of a site completes a form.

After the site user has filled out the form and submitted the data, they fall into a special script on the server called the form handler. In the form handler, the data can be saved to a database or sent by email.

If the user has filled out the web form correctly, without errors, then after sending the data to the server, the user will see the page for successfully filling out the form or redirecting it to a predetermined page. If the user fills out a form for polling or voting, then after successfully filling out the form, he will immediately see a page with general voting results.

#### 1.2.2 Types of Web forms.

Feedback form or contact form is an html form that usually has several fields: name, e-mail, subject of the message, the text of the message itself and the button “Send”. After the form is filled in by the visitor of the site and presses the button “Send”, all the data from the form is sent to the site owner's e-mail. At the same time the site visitor will not have to open any additional email programs. In addition, using the feedback form, you do not disclose your e-mail address to spammers.

Feedback forms are one of the most effective modes of communication between customers and businesses. The customer can provide the the business with their experiences, requirements or suggestions. Different feedback forms are available for reaching out to customers.

HTML feedback forms are easier to design and construct with the help of software tools. Below presented figure 1.2 feedback form example.

The image shows a web form titled "Feedback Form!". It contains the following elements:

- Email\***: A text input field with a note: "We will send you an email describing how to activate your newsletter subscription."
- Name (optional)**: Two text input fields labeled "First" and "Last".
- Preferred newsletter format**: Two radio button options: "Text" and "Html".
- Verification\***: A reCAPTCHA widget showing a green checkmark and the text "I'm not a robot".
- Submit!**: A dark grey button at the bottom of the form.

Figure 1.2 – Feedback form example

Subscription form. Then send or place a link to the finished form or place it on your website as easy as a picture. After creating a form, you can, if you wish, create new or edit existing fields, as well as change the appearance of the form.

A subscription form is a form located on any page of the website or blog where the users can fill in the fields with their data to receive emails on topics related to their interests. The primary purpose of the subscription form is opting-in subscribers to the mailing list. By placing a subscription form on your website, you will provide your customers with the opportunity to subscribe to electronic mailings or to paper publications from your website.

In this case, you get a subscriber base, you can unload it at any time for mailings. All information received through the forms is stored in the system, and also sent to the email addresses you specify.

Allows simply request an e-mail, or you can ask the users for the parameters for additional filtering during mailing. For example, the rubric for which a person wants to subscribe, and perhaps his age and gender. And at any time unload the subscriber base and make the newsletter.

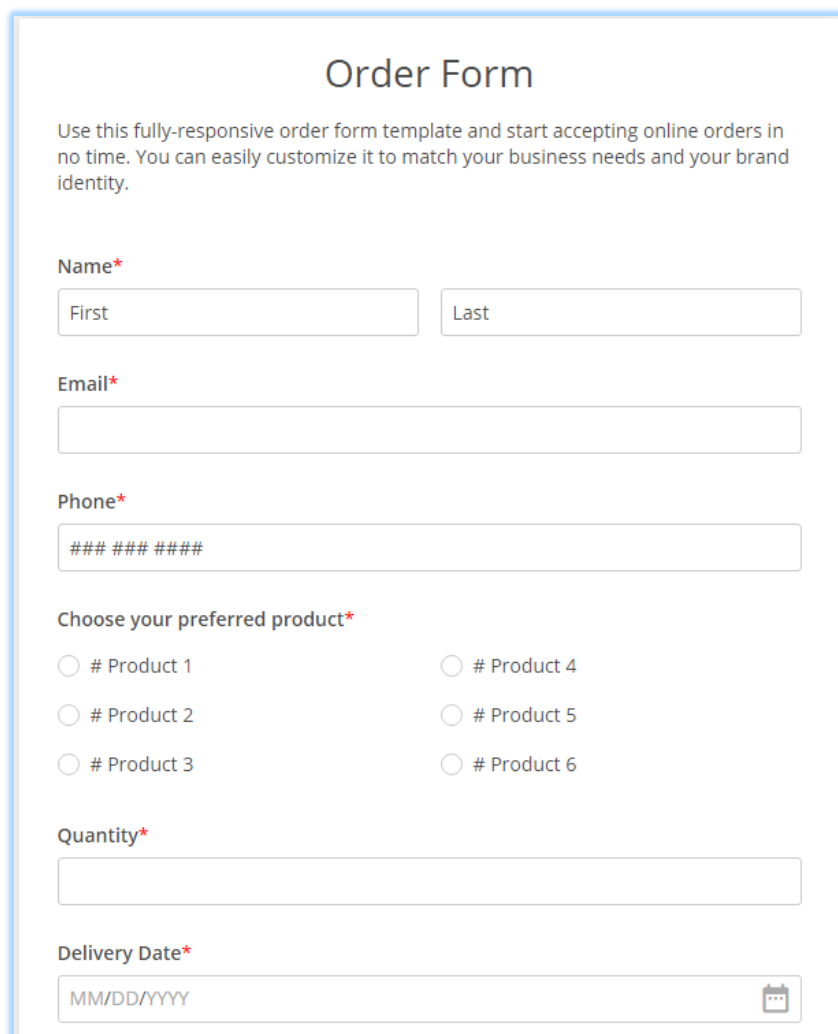
Order Form. The order form for goods is an ideal option for accepting orders for small online stores. You do not need to spend time and money on setting up and maintaining a complex online store. With the help of our online web form designer, you simply add the required fields to the order form, then paste the form code to your site and calmly accept orders.

All orders received from the form are stored in our system, and also sent to the e-mail addresses you specify, and can be exported from the system at any time in

CSV / Excel format for further processing. You can track the status of the order, as well as add service information on the order.

After creating the form, you can at any time add new fields or edit old ones, add protection against SPAM, set up validation rules and appearance.

Below in figure 1.3 has shown one example of order form.



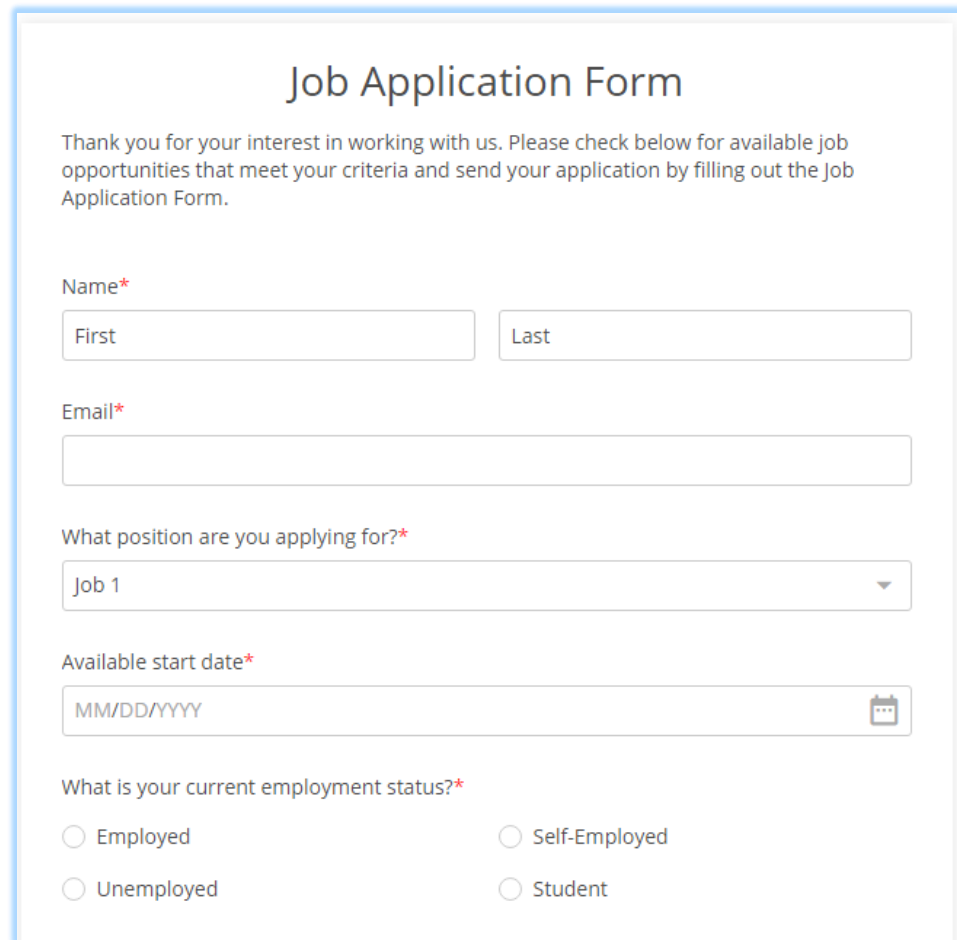
The image shows a web form titled "Order Form". Below the title is a descriptive paragraph: "Use this fully-responsive order form template and start accepting online orders in no time. You can easily customize it to match your business needs and your brand identity." The form contains several input fields: "Name\*" with sub-fields for "First" and "Last"; "Email\*" as a single line; "Phone\*" with a placeholder "### ### ####"; "Choose your preferred product\*" with six radio button options labeled "# Product 1" through "# Product 6"; "Quantity\*" as a single line; and "Delivery Date\*" with a date input field showing "MM/DD/YYYY" and a calendar icon.

Figure 1.3 – Order form example

Application form. You can accept applications for participation or registration in any event, whether it is an application for participation in a seminar or conference, or an application for a loan. All application forms are available online 24 hours a day, 7 days a week and 365 days a year. Therefore, your potential customers can apply at any time convenient for them, they will not even need to leave the site for this.

After creating the form, you can at any time add new fields or edit old ones, add SPAM protection, set up validation rules and appearance, as well as restrict access with a password. In figure 1.4 has shown job application form example.

All information received through the forms is stored in the system, and also sent to the email addresses you specify and can be exported from the system at any time in CSV / Excel format for further processing.



**Job Application Form**

Thank you for your interest in working with us. Please check below for available job opportunities that meet your criteria and send your application by filling out the Job Application Form.

**Name\***

First  Last

**Email\***

**What position are you applying for?\***

Job 1

**Available start date\***

MM/DD/YYYY

**What is your current employment status?\***

☐ Employed
 ☐ Self-Employed
 ☐ Unemployed
 ☐ Student

Figure 1.4 – Application form example

Testing form. In order to better know their visitors and subscribers, more and more site owners and mailing lists began to use the survey or testing form.

A survey or testing form resembles a voting form and may contain not one question, but several. In addition, in the survey form, as a rule, there are additional fields, for example, the name and contact information of the respondent.

Quiz can be called a new effective marketing tool with which you can identify the needs of the site visitor. Quiz is a step-by-step capture form that helps with each subsequent question more and more to interest the client, rather than immediately frighten him off with a questionnaire of 10-20 questions on one page, and finally leave a request and get his contact information.

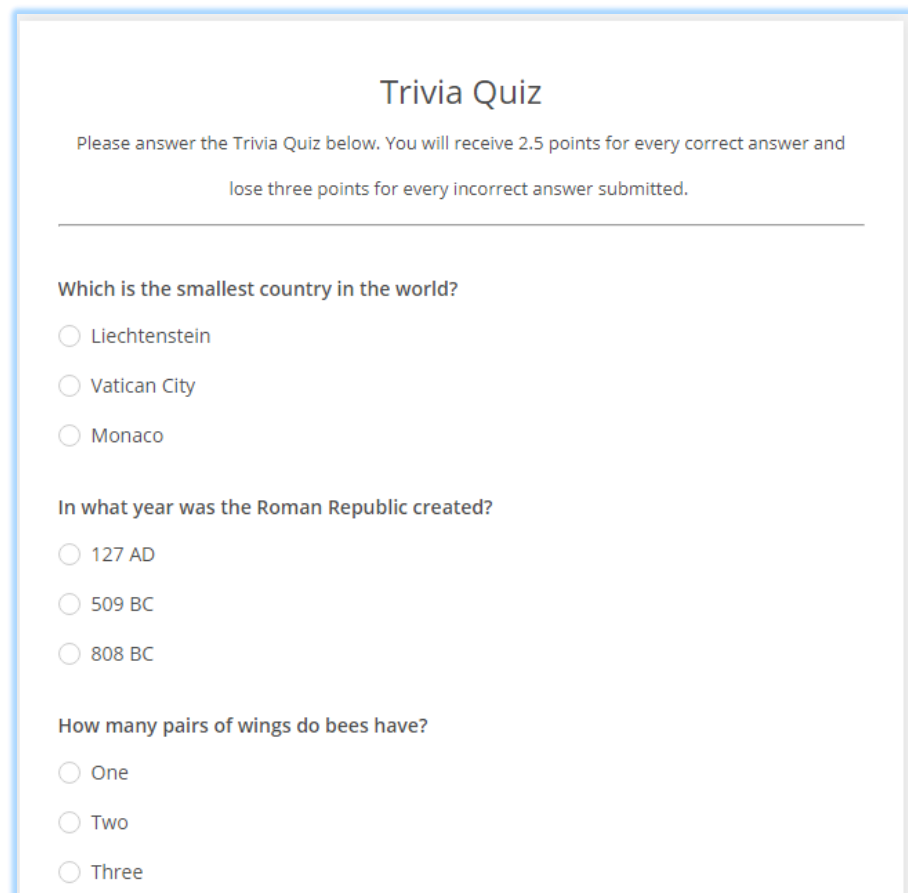
Quiz allows consistently more and more closer to bring the client to ensure that he left the application. Below in figure 1.5 presented example of quiz form.

Quiz or quiz-landing is implemented in a game form, where the visitor is asked a few questions step by step: Who is he? What does he want? How much is he willing to spend the money? When?

Quotes are actively used for educational and entertainment purposes, and now they have gained a pronounced trade effect. The quotes offered a decent, one might say, “softer” alternative to a more “assertive” filing of the request - to leave a request.

The site visitor answers questions with interest, since he knows that in the end he will find out, for example, what kind of discount he will receive on a product or service.

Voting form. In order to better know their visitors and subscribers, more and more site owners and mailings began to use the form for voting.



The image shows a web form titled "Trivia Quiz". Below the title, there is a paragraph of instructions: "Please answer the Trivia Quiz below. You will receive 2.5 points for every correct answer and lose three points for every incorrect answer submitted." A horizontal line separates the instructions from the questions. The first question is "Which is the smallest country in the world?" with three radio button options: "Liechtenstein", "Vatican City", and "Monaco". The second question is "In what year was the Roman Republic created?" with three radio button options: "127 AD", "509 BC", and "808 BC". The third question is "How many pairs of wings do bees have?" with three radio button options: "One", "Two", and "Three".

Figure 1.5 – Quiz form example

As a rule, one question and several answers are selected for voting. The user is prompted to select one or more options. After the user has voted, the average rating is calculated for each answer option.

### 1.3 Setting system tasks:

1. Research and find areas where using web-applications constructor.
2. Analyzing existing web-forms constructor. Comparing them and find useful elements of this web applications. Making decisions about existing systems functionality and their strong sides.
3. Using the necessary and effective tools and software. Use a lot technologies to easy and effective creating, development programming side single page application.
4. Development of web-application constructor. Which allows creating easy and quickly web-forms for own solutions.



## 2 Methods of implementing Information System

### 2.1 Single page application. Advantages of single page application.

Single Page Application - abbreviated SPA. In other words, a SPA is a web application hosted on a single web page that loads all the necessary code to work.

Along with loading the page itself. An application of this type appeared relatively recently, with the beginning of the HTML5 era. SPA is a typical representative of HTML5 applications.

If the application is quite complex and contains rich functionality, such as the distance learning portal, the number of files with scripts can reach several hundred or even thousands. To solve the problem of loading a large number of scripts in the SPA called API called AMD. AMD realizes the ability to download scripts on demand. That's is, if the initial page of a one-page portal was required three scripts, they will be loaded immediately before the program starts. What if when a user clicks on another page of a one-page portal, then AMD principle will load the script and markup just before going to this page.

The page of the site that contains all the links to all the CSS, and links for the scripts necessary for the operation of the SPA, it is commonly called "index.html". But pages that a user switches within a one-page portal called "modules". Consider the pros and cons of this approach. Below in figure 2.1 has shown difference between single page application and any regular website.

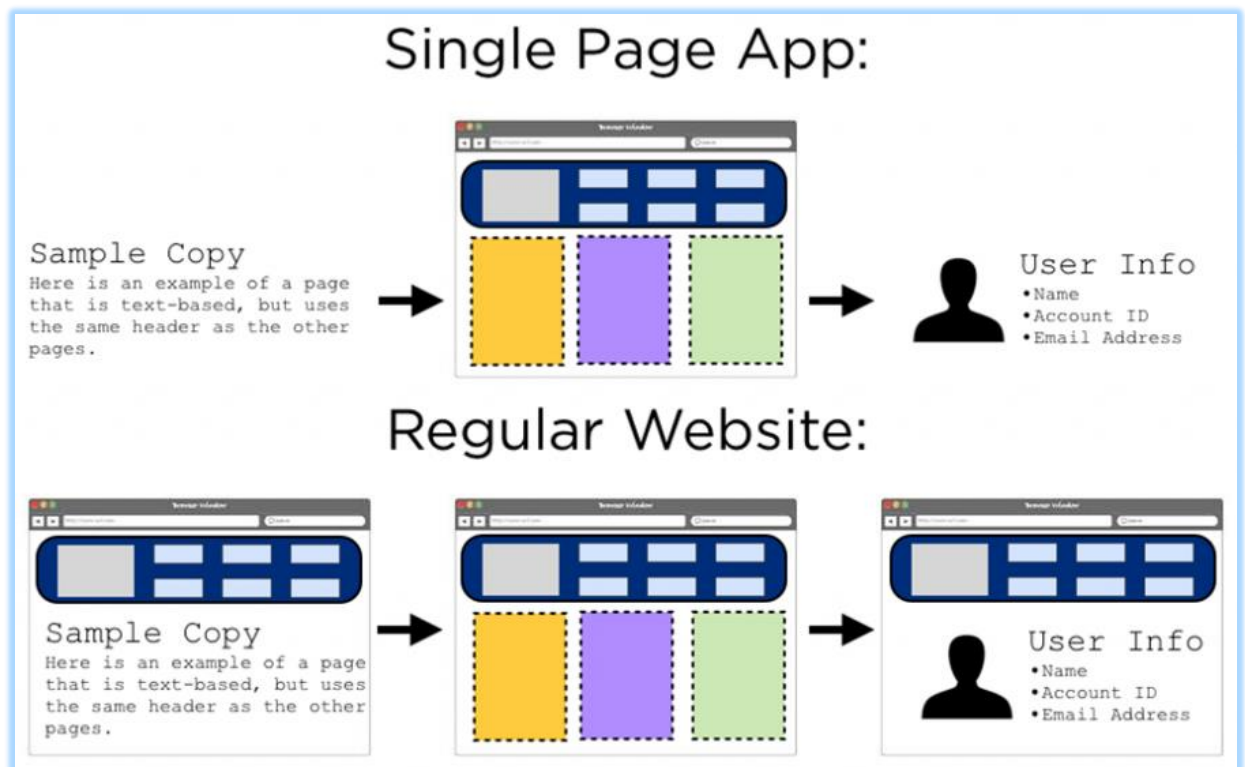


Figure 2.1 – Difference of Single page application and regular website

The disadvantages of using SPA are:

- the need to learn JavaScript. Creating a SPA involves a great use of javascript that leaves an imprint on the architecture applications;
- duplication of controllers and models both on the server and on the client. If you want to fix something, then it will be necessary to do it at least in two places;
- adding new functionality slows down the work. You need to control the client and server versions;
- complexity in testing.

And the below in figure 2.2 opposite of above has presented advantages of single page application.

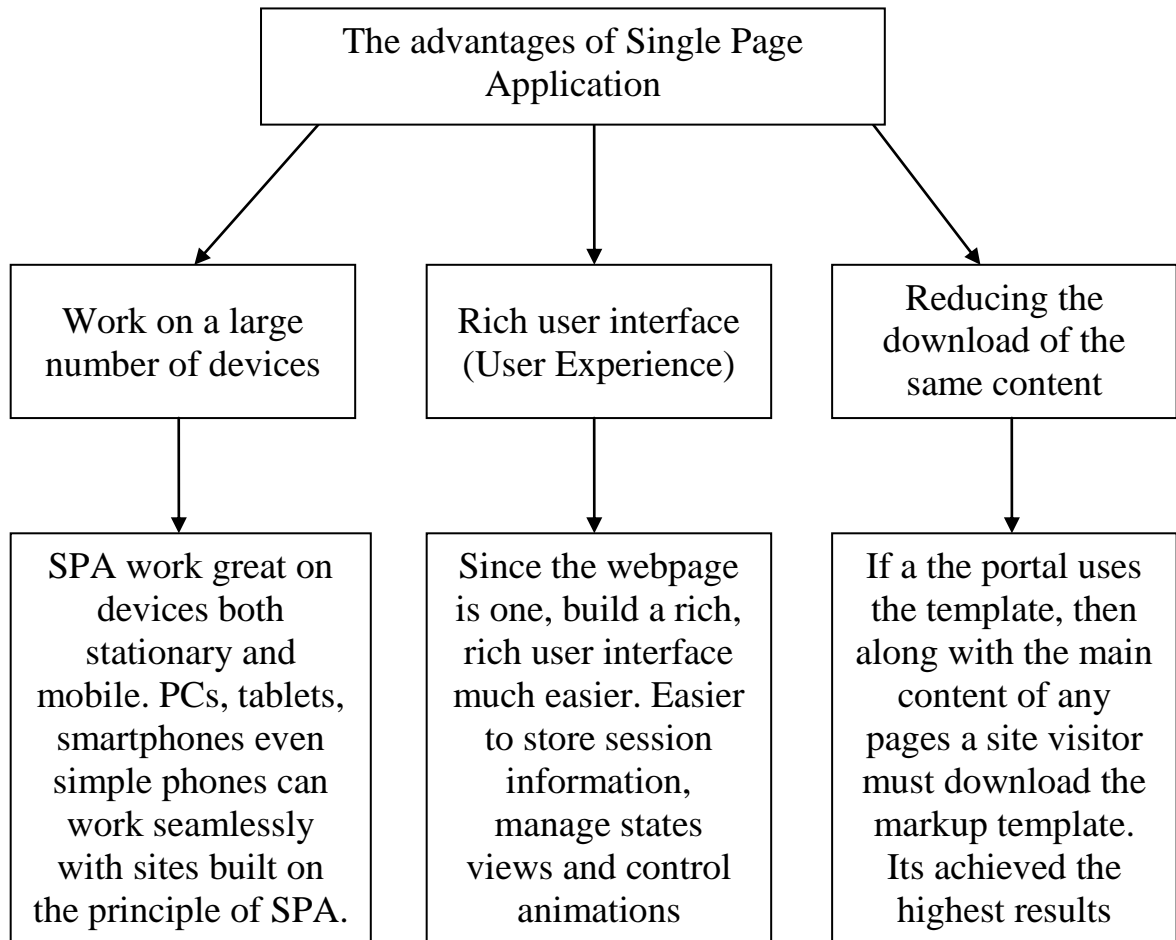


Figure 2.2 – The advantages of Single Page Application

Principles of any framework that implements the SPA paradigm must adhere to the following concepts and definitions:

- SPA supports client navigation. All user movements on the module pages are uniquely recorded in the navigation history, the navigation is “deep”, that is, if the user will copy and open the link to the internal module page in another browser or window; it will go to the corresponding page;

- SPA is located on one web page, it means everything you need for portal work scripts and styles should be defined in one place of the project - on a single web page;
  - SPA always keeps the client's work status in the browser's cache or in web storage;
  - SPA loads all the scripts required to start the application when web page initialization;
  - SPA gradually loads modules on demand.
- Examples of applications built on the principles of Single PageApplication is a lot. One of the most powerful and well-known is GMail - Google's postal service [4].

## **2.2 Source code editor Visual Studio Code**

Visual Studio Code is a lightweight yet powerful source code editor that runs on your desktop and is available for Windows, MacOS, and Linux. It comes with built-in support for JavaScript, TypeScript and Node.js and has a rich ecosystem of extensions for other languages (such as C ++, C #, Python, PHP, Go).

Visual Studio Code in action:

- intelligent code completion. Code smarter with IntelliSense - completion for variables, methods, and imported modules;
- optimized debugging. Debugging output is a thing of the past. Debug in VS code with your terminal tools;
- fast, powerful editing. Code analyzer, multi-cursor editing, parameter hints and other powerful editing features;
- code navigation and refactoring. Quickly review the source code by looking at the menu and navigating to the definition.

Visual Studio Code is a source code editor. It supports a number of programming languages, syntax highlighting, IntelliSense, refactoring, debugging, code navigation, Git support, and other features. Many of the features of the Visual Studio Code are not accessible through a graphical interface, they are often used through a palette of commands or JSON files (for example, user settings). The command palette is a command line similarity, which is invoked by a keyboard shortcut.

Visual Studio also allows you to replace the code page when saving a document, newline characters, and the programming language of the current document. In figure 2.3 presented user environment and interface Visual Studio Code.

Since 2018, the open source Python extension for Visual Studio Code has appeared. It provides developers with ample opportunities for editing, debugging and testing code. As of March 2019, through the built-in user interface in the product, you can download and install several thousand extensions only in the category "programming languages". Visual Studio Code is a code editor that supports work with more than 30 programming languages and file formats, including C #,

TypeScript, JavaScript. Not just a code editor, but a useful developer tool with extra features [5].

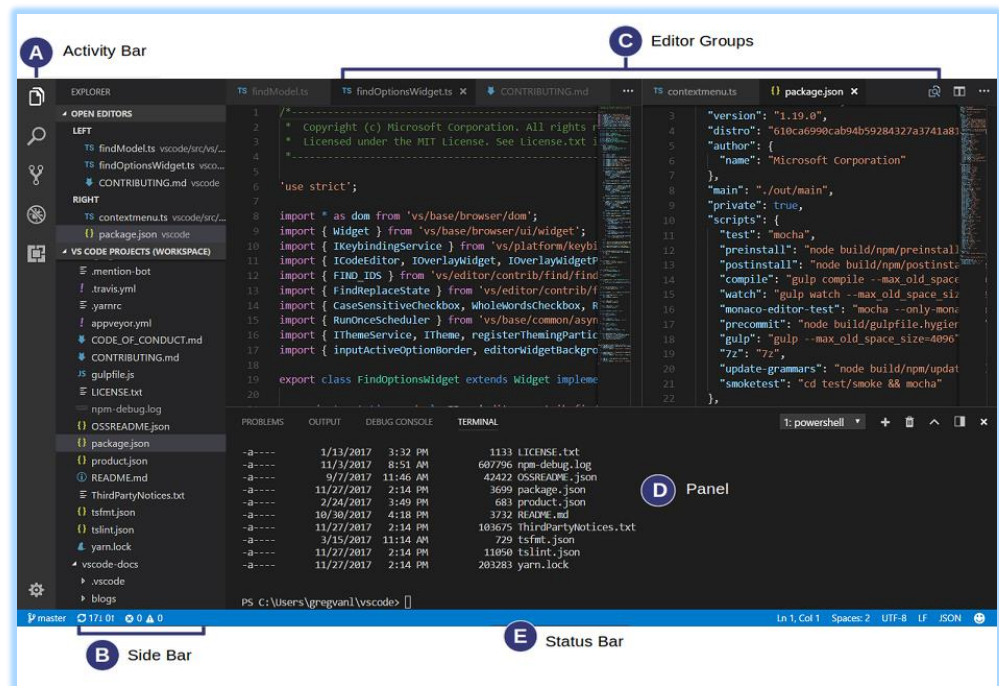


Figure 2.3 – User interface in Visual Studio Code

Visual Studio Code collects usage data (telemetry) and sends it to Microsoft. Although the provision of data is not mandatory and you may refuse to transfer personal data, some features, such as personalization, using such data will not be available to you to disable. Data may be transferred to Microsoft controlled subsidiaries, subsidiaries, and law enforcement agencies in accordance with the privacy statement.

Visual Studio Code characteristics:

- free open source text editor;
- it has IntelliSense (but it does not work immediately after installation, if Visual Studio is not installed, you must configure it to specify MinGW, etc.);
- smaller download size and RAM requirements. IntelliSense requires about 300 MB of RAM;
- works on younger computers. (startup is still slow, especially if PowerShell is used instead of CMD);
- lower support (open source, so you can change it yourself);
- build tasks depend on the project. Even if you want to build it in a vanilla configuration;
- mainly used for web development (this applies to all free text editors). They tend to brag about javascript/html support over C/C ++. Visual Studio demonstrates Visual Basic/C ++ in other languages;
- lack of good extensions (this is still new, though);
- it is difficult to reconfigure your project / workspace settings;

- cross-platform;
- it has a built-in terminal (PowerShell is too slow on startup);
- this is best for small projects and test code.

VS Code can be used on computers running Windows, OS X and Linux. The tool was released in the spring of 2015, and constantly updated. During the existence of Visual Studio Code has expanded its functionality, the list of supported languages based on feedback and suggestions from users.

The editor is based on open source products, which is sometimes an important criterion for developers, supports integration with version control systems, a built-in debugger and the ability to connect external tools [6].

## 2.3 Node.js software platform

Node.js is a software platform based on the V8 engine (developed by Google for the Chrome browser that translates JavaScript into native code), which transforms JavaScript from a highly specialized language into a general-purpose language. Node.js adds the ability of JavaScript to interact with I / O devices through its API (written in C ++), to connect other external libraries written in different languages, providing calls to them from JavaScript code.

Node.js is used mainly on the server, acting as a web server, but it is possible to develop desktop applications on Node.js (using NW.js, AppJS or Electron for Linux, Windows and macOS) and even program microcontrollers (for example, tessel and espruino). In figure 2.4 has shown many way to build an api server (REST api) with NodeJS. It could be matched to online database store likes MongoDB or MySQL.

Node.js is based on event-oriented and asynchronous (or reactive) programming with non-blocking I / O [7].

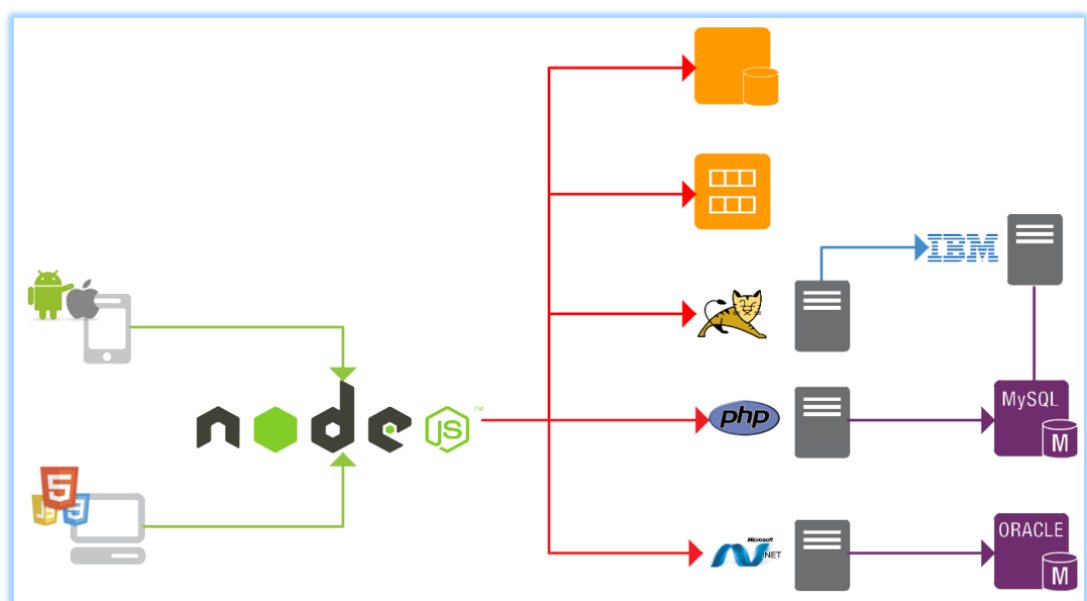


Figure 2.4 – Node.js application programming interface server

The main feature of Node.js is a software platform below:

- it is simple. Node.js is really very easy to learn, but you should first understand JavaScript itself, especially its asynchronous concepts;
- it is asynchronous. JavaScript runs in one thread, using events and callback functions to unload it. It was cool on the front end, it is still cool on the server! Virtually all objects in Node.js inherit from the EventEmitter class, that is, they are able to work with events. Read more about it here. If you do not understand the very concept of asynchrony, take a look here. A huge number of modules and libraries. The npm package manager ensures the rapid development of the Node.js ecosystem. Now it has more than 500 thousand open-source packages, and new ones appear every day. In addition, Node.js has a smart standard library;
- good old javascript. The same one that millions of developers are already using at the front end. Now they can safely switch to the server-side without learning a fundamentally different tool. There are differences, of course. First of all, Node.js has no DOM, cookies and other browser APIs. But there are many own useful methods and full control over the code execution environment. Here it is possible to use the most modern features of the language without fear and Babel, without looking at the limitations. The import system is also different. Browsers are starting to embed ES6 modules, and Node.js platform uses CommonJS with its require;
- V8 engine. This is an open source project written in C ++, which is being actively developed and improved by the efforts of thousands of developers. This has long been an adult serious language that can work for several hours in a row, so it makes sense to create ready-made compiled code. Modern engines combine interpretation and JIT compilation (just in time), which makes them very fast;
- node.js platform under the hood. Without going into the subtleties of the platform, let's take a look at its main parts [8].

Below has shown figure 2.5 Node.js architecture.

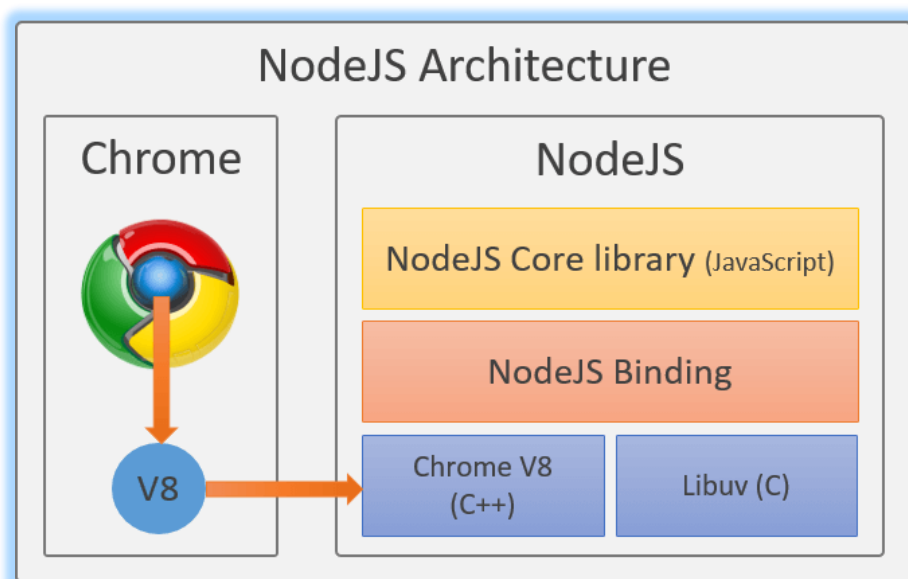


Figure 2.5 – Node.js architecture

And here you will find a small tour for beginners under the hood Node.js. Node.js platform can be installed in several ways: official installation files for different operating systems; package manager of the operating system; nvm.

There are also several options for deploying applications:

- create a local tunnel using ngrok or localtunnel;
- prototyping and demonstration sites: Glitch, Codepen;
- FAAS - serverless publishing using Serverless Framework or Standard

Library;

- PAAS solutions for every taste: Zeit, Heroku, Azure, Google Cloud;
- dedicated virtual server: Digital Ocean, Linode, Amazon Web Services;
- bare-metal server.

The event loop is a mechanism that accepts callback functions and registers them for execution at a certain point in the future. It works in the same stream as the JavaScript code itself. When an operation blocks a thread, the event loop also blocks.

A pool of workers is a performance model that calls and processes individual threads. Then they synchronously perform the task and return the result to the event loop. After the loop calls the callback function with the specified result.

In short, the pool of workers can deal with asynchronous I/O operations — first of all, we interact with the system disk and the network. This performance model is mainly used by modules like fs (demanding I / O speed) or crypto (demanding on CPU).

A pool of workers is implemented in libuv, which results in a small delay whenever Node requires a connection between JavaScript and C ++, but this delay is barely perceptible [9].

## **2.4 JavaScript library ReactJS for creating interfaces**

ReactJS - open JavaScript library for creating interfaces, which is intended to solve the problems of partial updating of the contents of the web page, which are often encountered in the development of one-page applications. Developed by Facebook, Instagram and community individual developers.

React allows developers to create large web applications. That use data that changes over time, without rebooting pages. His goal is to be fast, simple, scalable. React handles only the interface in applications.

Elements are JavaScript objects that are HTML elements. They do not exist in the browser, they describe DOM elements, such as a div, h1 or button. Components are React elements written by the developer. Usually these are parts of the user interface that contain their structure and functionality. For example, such as NavBar, LikeButton, or ImageUploader.

Properties - component options. They are provided as component arguments and look the same as HTML attributes. A state is a special object inside a component. He keeps data that may change over time. Composition - a combination of smaller components with the formation of more.



### Advantages of ReactJS:

- one-way data transfer. Properties are transferred to the component renderer as properties of the html tag. The component cannot directly change properties. However, the component may have an internal state. This is the this.state object, accessible within the component;
- virtual DOM.

React supports a virtual DOM (Document Object Model). It allows the library to determine which parts of the DOM have changed compared with the saved version of the virtual DOM, and thus determine how most efficiently update your browser DOM [11].

In figure 2.6 presented relationship between web page, real DOM and ReactJS virtual DOM.

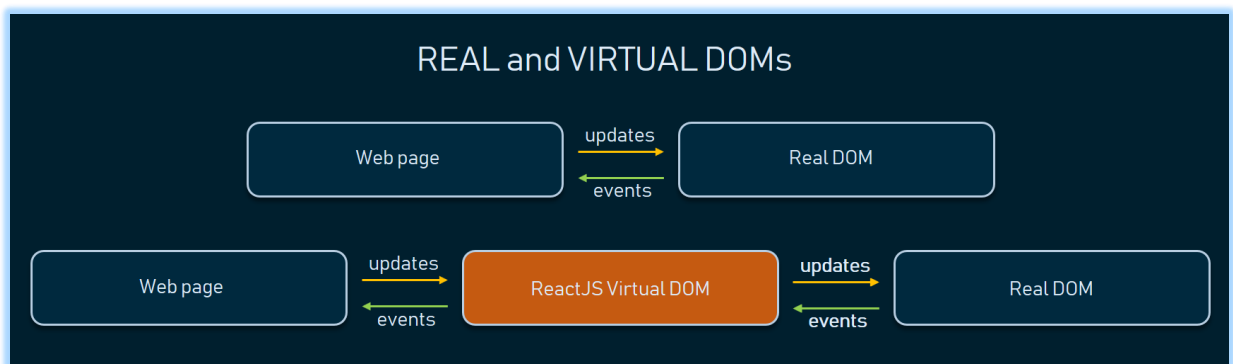


Figure 2.6 – React and Virtual DOMs

React components are usually written in JSX - syntax extensions JavaScript, similar to XML, which allows the use of syntax HTML tags for rendering components written in JSX compiled into calls to React library methods. Redux is an open source JavaScript library, designed to manage the state of the application. He basically used in conjunction with React to create user interfaces. Good for single-page applications where condition management can become difficult over time.

### Three basic principles of Redux:

- the only source of truth. Redux uses only one repository for the entire state. applications. Since the state is in one place, it calls the only source of truth. The data structure of the entire storage depends on the developer, but for a real application, this is usually an object with several levels of nesting;
- the status is read only. According to the Redux documentation, "The only way to change state - to transfer the action - an object that describes what happened. " it means that the application cannot directly change the state. Instead of this, It is necessary to transfer "action" to express the intention to change the state in repository;
- changes are made by "pure" functions. Redux does not allow state changes directly. Instead, action describes what changes need to be made. Reducers (reducers) are functions that handle actions and can make changes to state.

Reducers must be implemented as “pure” functions (pure functions), a term describing functions that satisfy the following conditions:

- they should not make external calls over the network or database;
- they return a value depending only on the transmitted parameters;
- their arguments are immutable, i.e. functions should not them change;
- a call to a pure function with the same arguments always returns same result;
- these functions are called “clean” because they do nothing only return a value depending on the parameters. They are independent of any part of the system.

The underlying concept of ReactJS is reusable components. The developer creates small parts of the code that can be combined to form larger ones or to use them as independent interface elements [12].

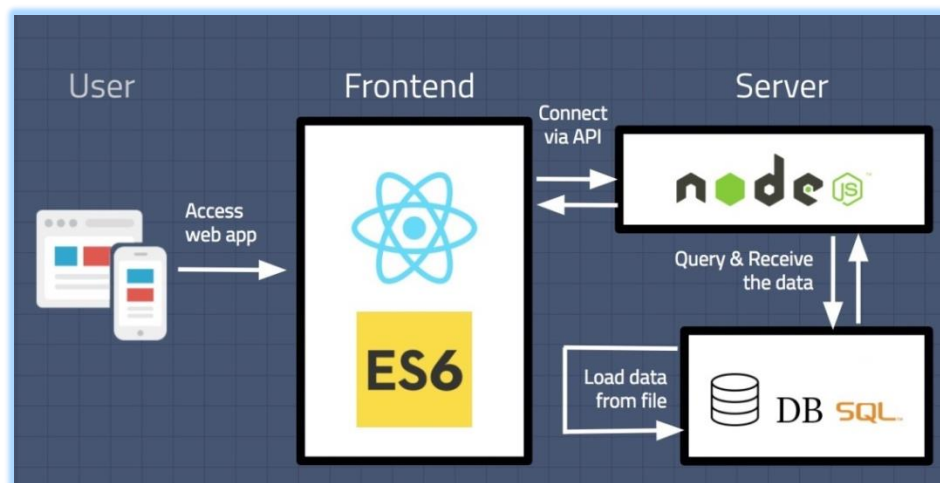


Figure 2.7 – User, ReduxJS and Server Node js connection

The second argument in ReactDOM.render is the document element that React will work with. In figure 2.8 presented ReactJS simple code example.

```
1 import React, { Component } from 'react';
2 import styled from 'styled-components';
3
4 const Wrapper = styled.div`
5   background: black
6 `
7
8 const Title = styled.h1`
9   color: white
10 `
11
12 class App extends Component {
13   render() {
14     return (
15       <Wrapper>
16         <Title>Hello Styled Components!</Title>
17       </Wrapper>
18     )
19   }
20 }
21
22 export default App;
```

Figure 2.8 – ReactJS simple code example

The names of the components begin with a capital letter. This is important, since the work will combine HTML elements and elements of React. Lowercase letters are reserved for HTML. If you try to name an element just a button, when rendering, the framework will ignore it and draw a regular HTML button.

Each element has a list of properties (attributes), as in HTML. In React, this is called props. The render function accepts the so-called JSX - this is HTML, placed in JavaScript and diluted with a special syntax. When processing events, it is important to understand that all the attributes of the React elements are named with camelCase. When working with functions, we pass the actual function reference, not the string. React.js creates a wrapper for the DOM event in the form of its own object in order to optimize the performance of working with events. Inside the handler, it is still possible to access all the methods available to the document.

First, the React.js template is defined to create elements from the component. Indicates where it will be used. For example, inside a call to the render function of another component or using ReactDOM.render.

The reaction creates an instance of the element and passes it a set of properties (props), access to which will be available through this.props. These properties are what we conveyed in the second step.

Since the above is a JavaScript, the class constructor method (if defined) will be called. React handles the result of the render function call.

## **2.5 ASP.NET Core 2.1/2.2 framework**

The ASP.NET Core platform represents technology from Microsoft, designed to create various kinds of web applications: from small websites to large web portals and web services.

On the one hand, ASP.NET Core is a continuation of the development of the ASP.NET platform. But on the other hand, this is not just another release. The release of ASP.NET Core actually means a revolution of the whole platform, its qualitative change. Development over the platform began in 2014. Then the platform was conditionally called ASP.NET vNext. In June 2016, the first release of the platform was released. And in May 2018, ASP.NET Core 2.1 was released, which is actually covered in the current tutorial.

ASP.NET Core is now fully open source framework. ASP.NET Core can run on top of a cross-platform .NET Core environment that can be deployed on the main popular operating systems: Windows, Mac OS X, Linux. And thus, with the help of ASP.NET Core we can create cross-platform applications. And although Windows as an environment for developing and deploying an application still prevails, but now we are not limited only to this operating system. That is, we can run web applications not only on Windows, but also on Linux and Mac OS. And to deploy a web application, you can use the traditional IIS, or cross-platform web server Kestrel.

Although ASP.NET Core is primarily aimed at using .NET Core, the framework can also work with the full version of the .NET framework. Due to the

modularity of the framework, all the necessary components of a web application can be loaded as separate modules through the Nuget package manager. In addition, unlike previous versions of the platform, there is no need to use the System.Web.dll library.

ASP.NET Core includes the MVC framework, which integrates MVC functionality, Web API and Web Pages. In previous versions of the platform, these technologies were implemented separately and therefore contained a lot of duplicate functionality.

ASP.NET Core is characterized by extensibility. The framework is built from a set of relatively independent components. And we can either use the built-in implementation of these components, or expand them using the inheritance mechanism, or even create and use our components with our own functionality.

It also simplified dependency management and project configuration. The framework now has its own lightweight container for dependency injection, and there is no longer a need to use third-party containers, such as Autofac, Ninject. Although if desired, they can also continue to use.

The new HTTP pipeline, which is based on Katana components and the OWIN specification, is now used to process requests. And its modularity makes it easy to add your own components. To summarize, the following key differences between ASP.NET Core and previous versions of ASP.NET can be distinguished:

- new lightweight and modular HTTP request pipeline;
- ability to deploy the application both on IIS;
- using the .NET Core platform and its functionality;
- distributing platform packages via NuGet;
- Integrated support for creating and using NuGet packages
- single web development stack combining Web UI and Web API;
- configuration for simplified use in the cloud;
- built-in support for dependency injection;
- extensibility;
- cross-platform: the ability to develop and deploy ASP.NET applications on Windows, Mac and Linux;
- development as open source, openness to change.

These and other features and capabilities became the basis for a new programming model [13].

## **2.6 NoSQL database program MongoDB**

MongoDB implements a new approach to building databases, where there are no tables, schemas, SQL queries, foreign keys, and many other things that are inherent in object-relational databases.

Since dinosaur times, it has been common to store all data in relational databases (MS SQL, MySQL, Oracle, PostgreSQL). It was not so important whether the relational databases were suitable for storing this type of data or not.

Unlike relational databases, MongoDB offers a document-oriented data model, which makes MongoDB faster, more scalable, and easier to use.

But, even taking into account all the shortcomings of traditional databases and the dignity of MongoDB, it is important to understand that tasks are different and methods for solving them are different. In some situations, MongoDB will really improve the performance of your application, for example, if you need to store data that is complex in structure. In another situation, it would be better to use traditional relational databases. In addition, you can use a mixed approach: store one type of data in MongoDB, and another type of data - in traditional databases. Below is shown figure 2.6 MongoDB interface.

The entire MongoDB system can represent not only one database located on one physical server. The MongoDB functionality allows you to place multiple databases on multiple physical servers, and these databases can easily exchange data and maintain integrity.

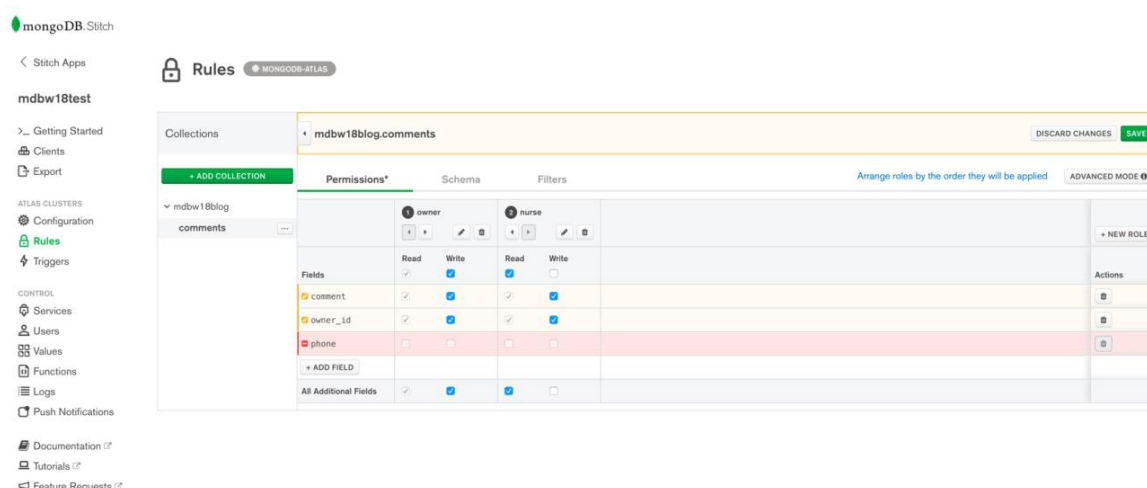
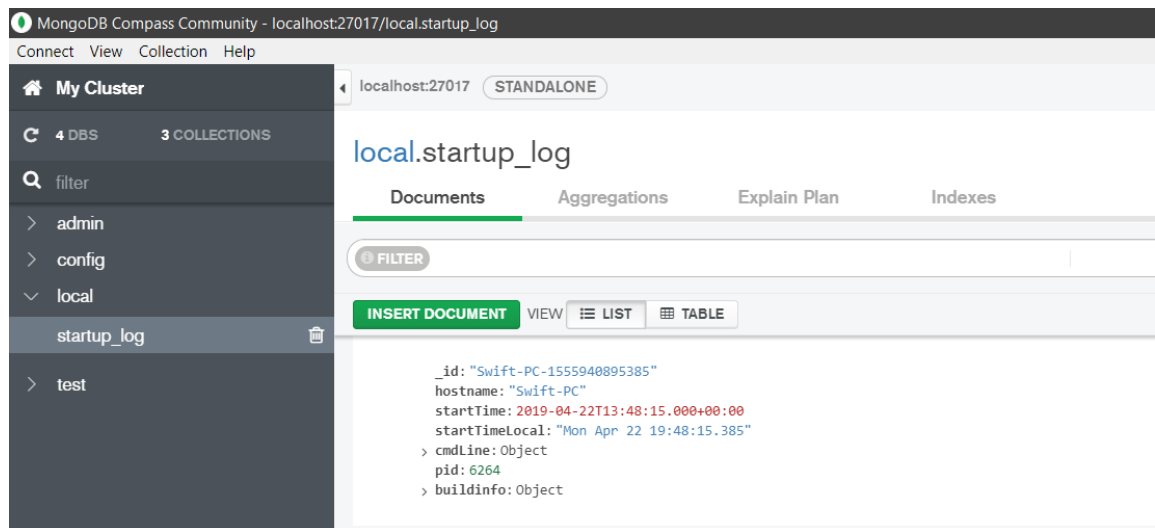


Figure 2.6– MongoDB interface

**Data Format in MongoDB.** One of the popular standards for data exchange and storage is JSON (JavaScript Object Notation). JSON effectively describes complex data structure. The way data is stored in MongoDB in this regard is similar to JSON, although formally JSON is not used. For storage in MongoDB, a format called BSON is used, or an abbreviation for binary JSON. BSON allows you to work with data faster: faster searching and processing. Although it should be noted that BSON, unlike data storage in JSON format, has a slight drawback: in general, data in JSON format takes up less space than in BSON format, on the other hand, this disadvantage is more than compensated for by speed. New shown in figure 2.7 saved as JSON format.

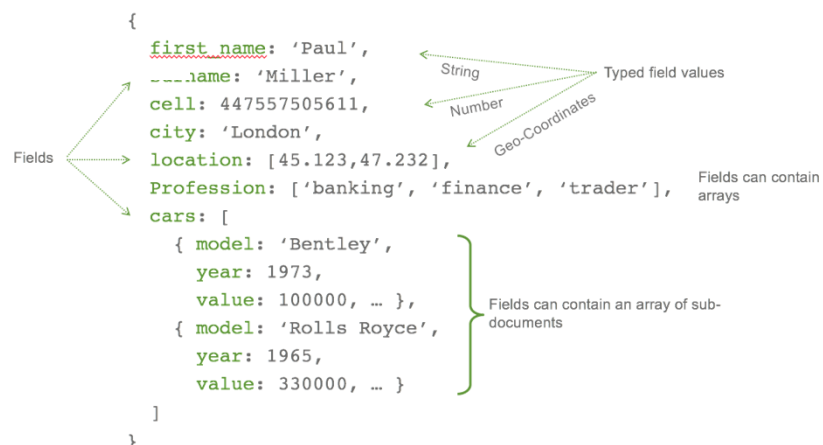
**Cross Platform.** MongoDB is written in C ++, so it is easy to port to a variety of platforms. MongoDB can be deployed on Windows, Linux, MacOS, Solaris platforms. You can also download the source code and compile MongoDB yourself, but it is recommended to use libraries from offsite.



2.7-picture – MongoDB startup\_log JSON format

Documents instead of lines. If relational databases store strings, MongoDB stores documents. Unlike lines, documents can store information that is complex in structure. The document can be represented as a repository of keys and values.

The key represents a simple label with which a certain piece of data is associated. However, with all the differences, there is one feature that brings MongoDB and relational databases together. In relational DBMS, there is such a thing as a primary key. This concept describes a column that has unique values. MongoDB has a unique identifier for each document called `_id`. And if you do not explicitly specify its value, MongoDB will automatically generate a value for it.



2.8-picture – MongoDB with Node.js and Express

Each key is assigned a specific value. But here you also need to take into account one feature: if there is a clearly delineated structure in relational databases, where there are fields, and if a field does not matter, you can assign a NULL value (depending on the settings of a particular database). In MongoDB, everything is different. If a value is not associated with a key, this key is simply omitted in the document and is not used.

### 3 Realization software for Information system

#### 3.1 Structure of software realization

For realization of Web-form Constructor “FormBuilder” were created special software structure. In each The structure of the software is presented in 3.1-picture.

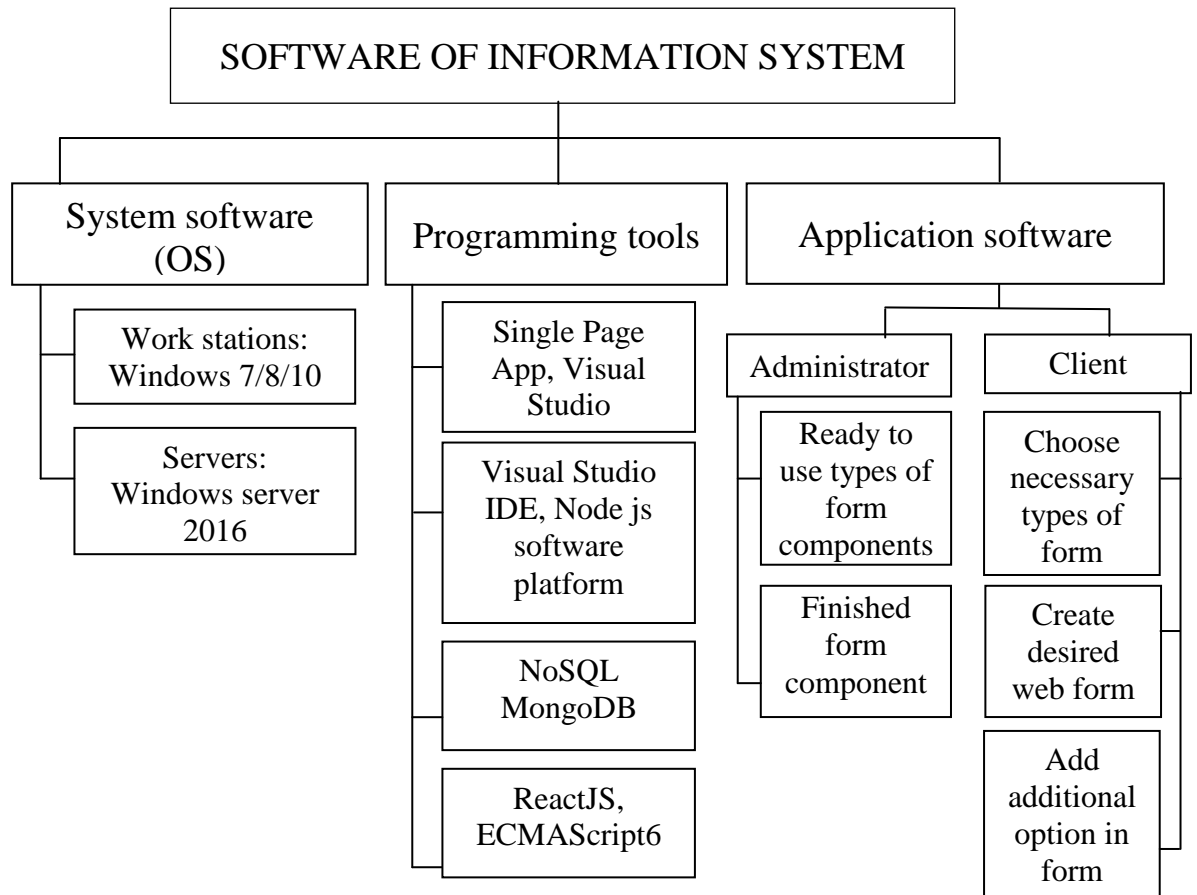


Figure 3.1 – Structure of software IS

It consists of:

- system software and OS;
- web-forms Constructor is implemented for Windows 7/8/10 and Linux workstations and web-browsers, such as, Chrome, Opera, Internet Explorer, etc;
- programming tools;
- the program is implemented in the JavaScript programming language used React open library on Node.js software platform, using Source code editor Visual Studio Code with Single Page App technologies. Query language NoSQL and Data Base Management System MongoDB.
- application software.

Web-forms Constructor on the administrator’s side includes:

- function “Prepare ready to use types of web-forms”. This function allows to create web-forms with ready to use types;



- function “Prepare finished form component”. With this function, Administrator of web-forms constructor can provide efficient state of web-forms for using.

Web-forms Constructor on the client side includes:

- function “Choose necessary type of web-form”. With this feature, client can choose necessary type and save time without creating a form right from the start;
- function “Create desired web-form”. Client can quickly create desired form with self-requirements and ready to using;
- function “Add additional option in form”. This feature allows client add some additional function for need web-form and can change in any moment of creating.

### **3.2 Justification of the choice of programming languages**

This diploma work is intended to work with web forms. The main task of the development of web forms Constructor for the use of end customers. The system should be easy to implement in various operating systems and web browsers. Therefore, a programming language must ensure efficient operation and control the content of a web page. To write a software product, a programming language and a scripting language JavaScript and a query language NoSQL are chosen. Also used stacks technology Node.js and library React.js.

Javascript is a simple and convenient language that allows you to easily manage the contents of a web page, tracking and responding to various user actions. Due to this, javascript has practically no competitors in its field of application and is the first language, the study of which you need to start a web developer.

JavaScript is a prototype-oriented scripting programming language. It is a dialect of ECMAScript. JavaScript is typically used as an embeddable language for programmatically accessing application objects.

The main architectural features: dynamic typing, weak typing, automatic memory management, prototype programming, functions as first-class objects.

After reviewing the most popular client-side languages, JavaScript was selected for the following reasons:

- cross-platform;
- does not require additional software (Flash Player for ActionScript, .NET Framework for Silverlight);
- javascript is an interpreted language, so to get the result you do not need to carry out the compiling process, it is enough to reload the web page in the browser.

Modern JavaScript is a “secure” general-purpose programming language. It does not provide low-level means of working with memory, processor, as it was originally focused on browsers in which it is not required.

As for the other features, they depend on the environment in which JavaScript is running. In the browser, JavaScript can do everything related to page manipulation, interaction with the visitor and, to some extent, with the server:

- create new HTML tags, delete existing ones, change element styles, hide, show elements, etc;
- respond to the actions of the visitor, handle mouse clicks, move the cursor, press the keyboard, etc;
- send requests to the server and upload data without reloading the page (this technology is called "AJAX");
- receive and set cookies, request data, display messages;
- full HTML / CSS integration.

JavaScript is the most common way to create browser interfaces.

The main reason for this is its lightness and ability to support full stack. All thanks to the popularity of the NodeJs & MEAN technology stack. Large technology companies are developing their product using javascript.

NodeJS is a software platform for many developers. This is a strong foundation for all JS programmers. NodeJS helps create both desktop and server applications in JavaScript, without the need for a browser.

React.js - a library from Facebook and Instagram, allows you to develop scalable applications that meet all modern requirements that are changing so quickly. ReactJS is reliable and stable. Simple interface design and implementation of a virtual DOM are key reasons for its popularity.

SQL databases are specifically designed for specific data models and have flexible schemas that allow you to develop modern applications. SQL databases are widely used due to ease of development, functionality and performance at any scale. They use various data models, including document, graph, search, using key-value pairs and storing data in memory.

Traditional DBMSs are guided by the requirements of the ACID for the transactional system: atomicity, consistency, isolation, durability.

### **3.3 Description of the developed application software**

3.3.1 General Information. This program is written using JavaScript React open library on Node.js software platform, using Source code editor Visual Studio Code with Single Page App technologies. Query language SQL and Data Base Management System Microsoft SQL Server 2017. For the normal functioning of the program necessary Windows 7/8/10 and Linux operating systems and web-browsers, such as, Chrome, Opera, Mozilla, Microsoft Edge, Internet Explorer, etc.

3.3.2 Functional purpose. This program is intended for implementation on a computer Web-forms Constructor “FormBuilder”. This system allows creating web-forms for any tasks and solutions.

3.3.3 Used technical tools. When developing and debugging this diploma work, the following technical tools were used:

- CPU: Intel(R) Core(TM) i5-7200U 2.50GHz;
- RAM: 8 Gb;
- Solid State Drive: 256 Gb;

- Removable FlashDrive disk 32 Gb USB 3.1 Type-C;
- Network adapter Intel wireless-AC9560;
- Video adapter: NVIDIA GeForce GTX 1060;
- printer Samsung dj 940c;
- standard keyboard and mouse.

3.3.4 Call and Download. The program is called by web-browser, such as Google Chrome. After that the path in the address bar is specified as <https://www.formbuilder.kz>. And this address provide using system web-form Constructor.

3.3.5 Input Data. The input data of the diploma work are structure of existing web-application constructor and web-form constructor.

3.3.6 Output Data. The output data of the diploma work ready to use and efficient web-forms.

### 3.4 Realization of Information system.Creating a single page application

3.4.1 Creating a Node.js Application in Visual Studio. Creating a Node.js and React application in Visual Studio. In Visual Studio, you can easily create a Node.js project and use IntelliSense and other built-in functions that support Node.js.

– NodeJS is a Server-side technology that helps generate HTML returned to the user's browser, it is similar to other technologies like PHP, Java Servlet / JSP, ... The difference here is that NodeJS uses Javascript to write code, so you only need to know Javascript and you can program applications from 2 sides of Client & Server. React is an interface platform for creating user interfaces. JSX is a JavaScript syntax extension commonly used with React to describe user interface elements. JSX code needs to be converted to plain javascript code before it can be run in a browser. webpack integrates javascript files so that they can be run in a browser. It can transform or pack other resources and assets. It is often used to tell the compiler, such as Babel or TypeScript, to convert JSX or TypeScript code into regular JavaScript code. Below in figure 3.3 has shown how to start Node.js development in Visual Studio.

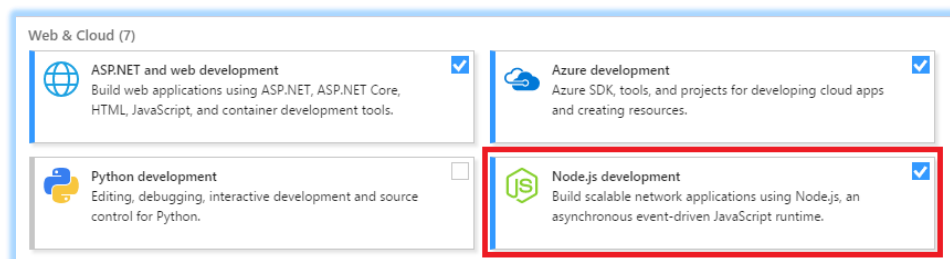


Figure 3.3 – "Node.js Development" workload

3.4.1 Create React App. The Create React App is a convenient environment for learning React and the best way to start creating a new one-page application on React. It sets up your development environment, so you can use the latest JavaScript

features, provide design convenience, and optimize your application for production. You will need Node  $\geq 6$  and npm  $\geq 5.2$ . To create a project, run. In figure 3.5 presented.

```

1: node
npm install -g create-react-app
C:\Users\eqvin\AppData\Roaming\npm\create-react-app -> C:\Users\eqvin\AppData\Roaming\npm\node_modules\creat
e-react-app\index.js
+ create-react-app@2.1.8
added 63 packages from 20 contributors in 6.821s
PS C:\Users\eqvin\Desktop\WebConstructor> create-react-app jwt-react

Creating a new React app in C:\Users\eqvin\Desktop\WebConstructor\jwt-react.

Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts...

yarn add v1.13.0
[1/4] Resolving packages...
[2/4] Fetching packages...
info fsevents@1.2.7: The platform "win32" is incompatible with this module.
info "fsevents@1.2.7" is an optional dependency and failed compatibility check. Excluding it from installati

```

Figure 3.5 – create-reat-app comand

Below has shown main pages in FormBuilder application in the figure 3.6, figure 3.7 and figure 3.8. In the table 3.1 presented main form parametres.

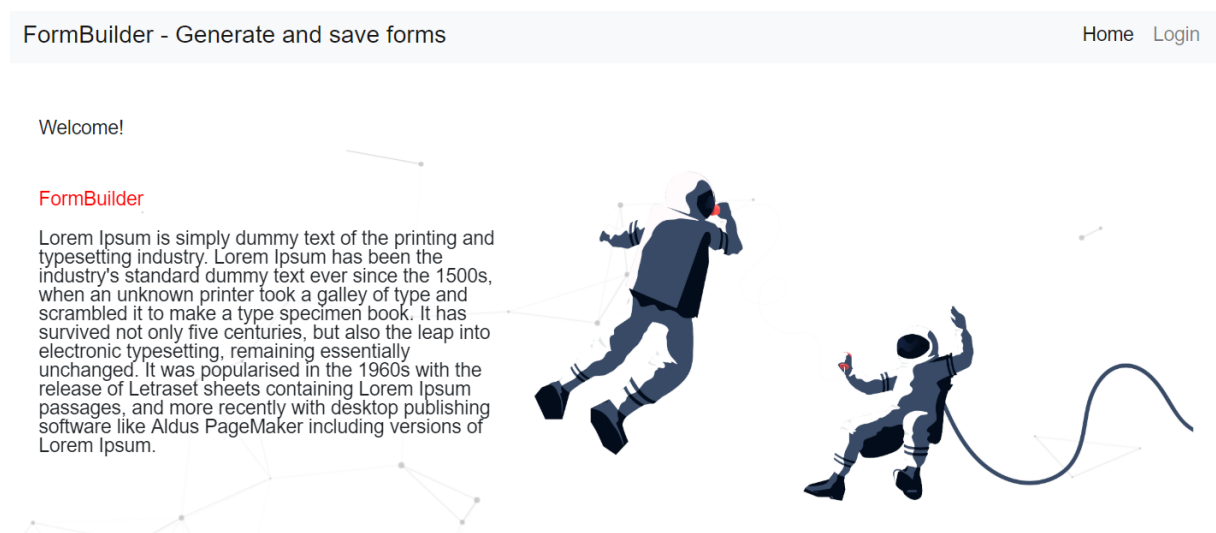


Figure 3.6 – Main page FormBuilder program

Table 3.1 – Forms parametres

| Nº | Name        | Type     | Description  |
|----|-------------|----------|--|
| 1  | form_action | string   | URL path to submit the form                            |
| 2  | action_name | string   | Defines form submit button text. Defaults to "Submit"  |
| 3  | form_method | string   | Verb used in the form submission.                      |
| 4  | onSubmit    | function | Invoke submit data, if exists will override form post. |
| 5  | back_action | string   | URL path to go back if needed.                         |
| 6  | variables   | object   | Key/value object for Signature variable replacement.   |

Preview
Alternate/Short Form
Preview Form

### Scholarship Form

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book.

How did you hear about this scholarship? Required

Internet

How hard is the material?

Easy

12345

Difficult

What is your favorite subject?

☐ Math
 ☐ Physics
 ☐ English
 ☐ Art

Signature of Applicant Required

**Toolbox**

H

Header Text

A

Label

¶

Paragraph

↔

Line Break

▢

Dropdown

🏷

Tags

☑

Checkboxes

⦿

Multiple Choice

A

Text Input

+

Number Input

T

Multi-line Input

🖼

Image

★

Rating

📅

Date

✍

Signature

Figure 3.7 – Form builder drag and drop form generator

Dropdown

Display Label

How did you hear about this scholarship?

☒ Required

Print Options

☐ Page Break Before Element?

Alternate/Short Form

☒ Display on alternate/shorter form?

Options

Internet

☐

+

Other

☐

+

-

Option text

☐

+

-

Value

Correct

**Toolbox**

Header Text

Label

Paragraph

Line Break

Dropdown

Tags

Checkboxes

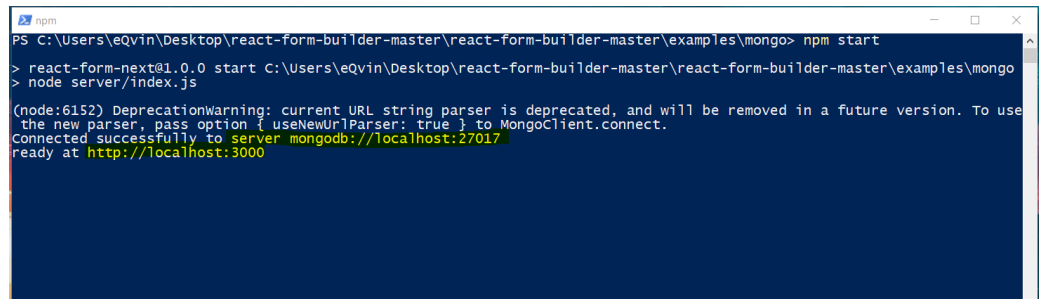
Multiple Choice

Text Input

Number Input

Figure 3.8 – Custom form example

Below in figure 3.9 has shown how to run the program in PowerShell



```
PS C:\Users\eqvin\Desktop\react-form-builder-master\react-form-builder-master\examples\mongo> npm start
> react-form-next@1.0.0 start C:\Users\eqvin\Desktop\react-form-builder-master\react-form-builder-master\examples\mongo
> node server/index.js

(node:6152) DeprecationWarning: current URL string parser is deprecated, and will be removed in a future version. To use
the new parser, pass option { useNewUrlParser: true } to MongoClient.connect.
Connected successfully to server mongodb://localhost:27017
ready at http://localhost:3000
```

Figure 3.9 – Run program with comand npm start

Next in figure 3.10 presented how to connect to MongoDB. Server MongoDB: localhost:27017 and program ready at http: //localhost:3000

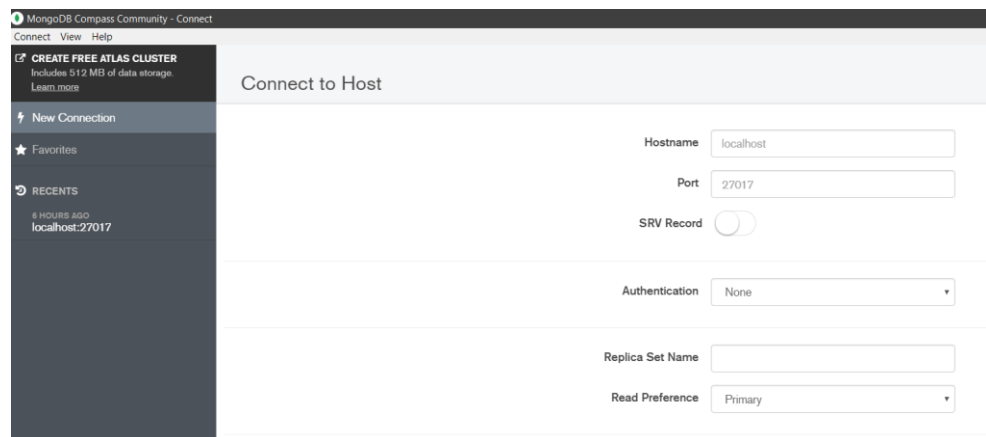


Figure 3.10 – MongoDB connection

Saved JSON data has shown in figure 3.9 components has saved.

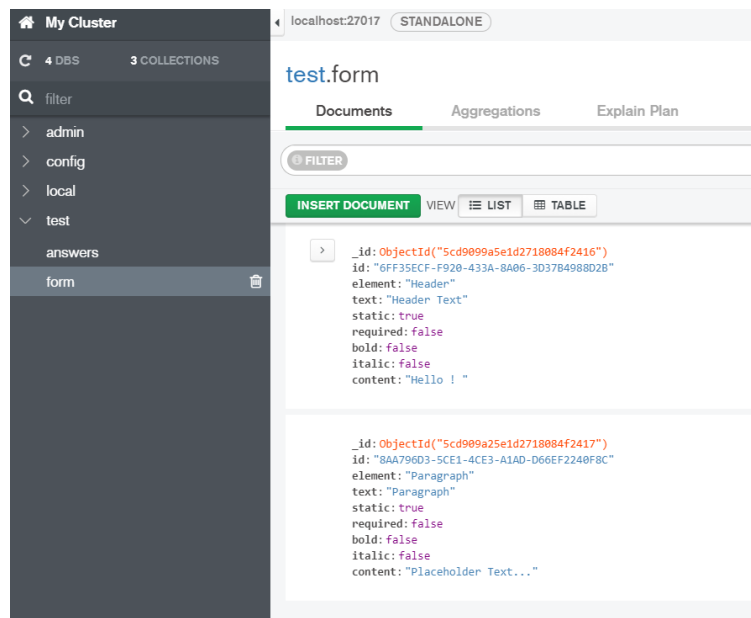


Figure 3.9 – MongoDB saved JSON



## CONCLUSION

Often, when creating websites, it becomes necessary to install a web form on a page (for example, it can be a user registration form, a feedback form, an online order, etc.). This is necessary to enable resource users to contact the site's administration send any data. Creating web forms will increase the functionality of the site as a whole.

Web-forms are one of the most important tools used by companies and organizations. With their help, you can get the opinions of users and conduct a variety of polls. Creating forms using a specially designed for this case constructor is much easier and faster. They are able to create for a few clicks useful and well-designed projects.

Constructor of Web-forms is a simple tool that allows you to save valuable time and in a few clicks create a modern form, the code of which can be copied and installed on external sites. Constructor allows you to visually describe the composition of the details that will be placed on the form, and choose the option of placing the command panel.

The first part of the work is devoted to describing the basics of working with web applications, especially, web forms. Researching the structure, elements and main functions of web forms. Also shows special types of web-forms, which often used.

The second part describes the Information System. The main technologies and tools for the implementation of the information system. Shows the effectiveness, advantages and disadvantages of each software tool. There is a rationale for the choice of platforms for the development and implementation of the program.

The third part is the implementation of the program. Demonstration of the main functionality and options of the web-forms constructor, by step-by-step program description. Using software tools such as: Single page application, Visual Studio Code, Node.js software platform, JavaScript library React.js.

## REFERENCES

- 1 Semmy P. Learning Web App Development / South-Western, Cengage Learning – 2014. – P. 106-168.
- 2 Douglas J. R. Programming Microsoft Web Forms (Developer Reference) / Wiley – 2006. – 298 p.
- 3 Emmit S. SPA Design and Architecture: Understanding Single Page Web Applications / Wrox – 2015. – 98 p. // Electronic version in web-site <https://www.amazon.com/SPA-Design-Architecture-Understanding-Applications/dp/1617292435>
- 4 Bruno J. D., Mithun S., Jason K. Web Development with Node Third Edition / Apress – 2017. – 72 p.
- 5 Caroline J., Gerry G., Steve K. Forms that Work: Designing Web Forms for Usability Interactive Technologies 1st Edition / Packt – 2008. – 295 p.
- 6 Alex M. JavaScript Web Applications / O'Reilly Media – 2011. – 295 p. // Electronic version in web-site <https://www.oreilly.com/library/view/javascript-web-applications/9781449308216/>
- 7 David H. Node Web Development Second Edition / Packt – 2011. – 35 p. // Electronic version in web-site <https://ru.scribd.com/book/253047202/Node-Web-Development>
- 8 Eric B. Full-Stack JavaScript Development: Develop, Test and Deploy Node / Red Hat Press – 2016. – P. 164-300.
- 9 Mike C., Marc H., TJ Holowaychuk, Nathan R. Node.js in Action 1st Edition / Manning – 2011. – 235 p. // Electronic version in web-site // <https://www.manning.com/books/node-js-in-action>
- 10 Japs Ph., Dewey B. Building Web Applications with Visual Studio / Apress – 2011. – P. 244-357.
- 11 Alex B., Eve P. Learning React: Functional Web Development / O'Reilly – 2017. – P. 164-300.
- 12 Prathamesh S. ReactJS by Example - Building Modern Web Applications with React / O'Reilly – 2016. – 125 p.
- 13 Eelco P., Peter M., David H. Core Basics / McGraw-Hill Education – 2013. – 301 p.
- 14 Deepak V. MS SQL Server for Development / McGraw-Hill Education – 2013. – 301 p. // Electronic version in web-site // <https://books.goalkicker.com/MicrosoftSQLServerBook/>
- 15 Stoyan S. React: Up & Running: Building Web Apps // O'Reilly Media – 2016. – P. 87-193.

## Appendix A

A complete react form builder that interfaces with a json endpoint to load and save generated forms.

The screenshot shows the main form builder interface. On the left, a 'Preview' tab displays a 'Scholarship Form'. The form includes a header, a paragraph of Lorem Ipsum text, a dropdown menu for 'How did you hear about this scholarship?' (with a 'Required' label), a slider for 'How hard is the material?' (ranging from 1 to 5), a radio button group for 'What is your favorite subject?' (Math, Physics, English, Art), and a signature field for 'Signature of Applicant' (with a 'Required' label). On the right, a 'Toolbox' lists various form elements: Header Text, Label, Paragraph, Line Break, Dropdown, Tags, Checkboxes, Multiple Choice, Text Input, Number Input, Multi-line Input, Image, Rating, Date, and Signature.

Figure A.1 – Main form builder page

The screenshot shows the editing interface for a 'Dropdown' element. The 'Display Label' is 'How did you hear about this scholarship?'. The 'Required' checkbox is checked. Under 'Print Options', 'Page Break Before Element?' is unchecked. Under 'Alternate/Short Form', 'Display on alternate/shorter form?' is checked. A table lists the options for the dropdown:

| Options     | Value                | Correct  |
|-------------|----------------------|--|
| Internet    | <input type="text"/> | <input type="checkbox"/> <input type="button" value="+"/>                                  |
| Other       | <input type="text"/> | <input type="checkbox"/> <input type="button" value="+"/> <input type="button" value="-"/> |
| Option text | <input type="text"/> | <input type="checkbox"/> <input type="button" value="+"/> <input type="button" value="-"/> |

The 'Toolbox' on the right lists the available form elements: Header Text, Label, Paragraph, Line Break, Dropdown, Tags, Checkboxes, Multiple Choice, Text Input, and Number Input.

Figure A.2 – Editing items in form builder page

## Appendix B

### Main codes in the application.

#### Basic usage

```
var React = require('react');
var FormBuilder = require('react-form-builder2');

React.render(
  <FormBuilder.ReactFormBuilder />,
  document.body
)
```

#### Form properties

```
var items = [{
  key: 'Header',
  name: 'Header Text',
  icon: 'fa fa-header',
  static: true,
  content: 'Placeholder Text...'
},
{
  key: 'Paragraph',
  name: 'Paragraph',
  static: true,
  icon: 'fa fa-paragraph',
  content: 'Placeholder Text...'
}
];

<FormBuilder.ReactFormBuilder
  url='path/to/GET/initial.json'
  toolbarItems={items}
  saveUrl='path/to/POST/built/form.json' />
```

#### React form generator

```
var React = require('react');
var FormBuilder = require('react-form-builder2');

React.render(
  <FormBuilder.ReactFormGenerator
    form_action="/path/to/form/submit"
    form_method="POST"
    task_id={12} // Used to submit a hidden variable with the id to the form from
the database.
    answer_data={JSON_ANSWERS} // Answer data, only used if loading a pre-
existing form with values.
    authenticity_token={AUTH_TOKEN} // If using Rails and need an auth token to
submit form.
    data={JSON_QUESTION_DATA} // Question data
  />,
  document.body
)
```

## Appendix B continuation

ReactJS code ~/nodeModules/lib/index.js

```
'use strict';
var _createClass = function () { function defineProperties(target, props) { for (var i = 0; i < props.length; i++) { var descriptor = props[i]; descriptor.enumerable = descriptor.enumerable || false; descriptor.configurable = true; if ("value" in descriptor) descriptor.writable = true; Object.defineProperty(target, descriptor.key, descriptor); } } return function (Constructor, protoProps, staticProps) { if (protoProps) defineProperties(Constructor.prototype, protoProps); if (staticProps) defineProperties(Constructor, staticProps); return Constructor; }; }();
var _react = require('react');
var _react2 = _interopRequireDefault(_react);
var _reactDnd = require('react-dnd');
var _reactDndHtml5Backend = require('react-dnd-html5-backend');
var _reactDndHtml5Backend2 = _interopRequireDefault(_reactDndHtml5Backend);
var _preview = require('./preview');
var _preview2 = _interopRequireDefault(_preview);
var _toolbar = require('./toolbar');
var _toolbar2 = _interopRequireDefault(_toolbar);
var _form = require('./form');
var _form2 = _interopRequireDefault(_form);
var _store = require('./stores/store');
var _store2 = _interopRequireDefault(_store);
function _interopRequireDefault(obj) { return obj && obj.__esModule ? obj : { default: obj }; }
function _classCallCheck(instance, Constructor) { if (!(instance instanceof Constructor)) { throw new TypeError("Cannot call a class as a function"); } }
function _possibleConstructorReturn(self, call) { if (!self) { throw new ReferenceError("this hasn't been initialised - super() hasn't been called"); } return call && (typeof call === "object" || typeof call === "function") ? call : self; }
function _inherits(subClass, superClass) { if (typeof superClass !== "function" && superClass !== null) { throw new TypeError("Super expression must either be null or a function, not " + typeof superClass); } subClass.prototype = Object.create(superClass && superClass.prototype, { constructor: { value: subClass, enumerable: false, writable: true, configurable: true } }); if (superClass) Object.setPrototypeOf ? Object.setPrototypeOf(subClass, superClass) : subClass.__proto__ = superClass; } /**
var ReactFormBuilder = function (_React$Component) {
  _inherits(ReactFormBuilder, _React$Component);
  function ReactFormBuilder(props) {
    _classCallCheck(this, ReactFormBuilder);
    var _this = _possibleConstructorReturn(this, (ReactFormBuilder.__proto__ || Object.getPrototypeOf(ReactFormBuilder)).call(this, props));

    _this.state = {
      editMode: false,
      editElement: null
    };
    return _this;
  }
  _createClass(ReactFormBuilder, [{
    key: 'editModeOn',
    value: function editModeOn(data, e) {
      e.preventDefault();
      e.stopPropagation();
      if (this.state.editMode) {
        this.setState({ editMode: !this.state.editMode, editElement: null });
      } else {
        this.setState({ editMode: !this.state.editMode, editElement: data });
      }
    }
  }], {
  }, {
```

## Appendix B continuation

```
key: 'manualEditModeOff',
  value: function manualEditModeOff() {
    if (this.state.editMode) {
      this.setState({
        editMode: false,
        editElement: null
      });
    }
  }
}, {
  key: 'render',
  value: function render() {
    var toolbarProps = {};
    if (this.props.toolbarItems) {
      toolbarProps.items = this.props.toolbarItems;
    }
    return _react2.default.createElement(
      'div',
      null,
      _react2.default.createElement(
        'div',
        { className: 'react-form-builder clearfix' },
        _react2.default.createElement(
          'div',
          null,
          _react2.default.createElement(_preview2.default, { files: this.props.files,
            manualEditModeOff: this.manualEditModeOff.bind(this),
            parent: this,
            data: this.props.data,
            url: this.props.url,
            saveUrl: this.props.saveUrl,
            onLoad: this.props.onLoad,
            onPost: this.props.onPost,
            editModeOn: this.editModeOn,
            editMode: this.state.editMode,
            variables: this.props.variables,
            editElement: this.state.editElement }),
          _react2.default.createElement(_toolbar2.default, toolbarProps)
        )
      )
    );
  }
}]);

return ReactFormBuilder;
}(_react2.default.Component);

var FormBuilderers = {};

FormBuilderers.ReactFormBuilder = (0,
_reactDnd.DragDropContext)(_reactDndHtml5Backend2.default)(ReactFormBuilder);
FormBuilderers.ReactFormGenerator = _form2.default;
FormBuilderers.ElementStore = _store2.default;
module.exports = FormBuilderers;
```



## Appendix C

Main code view the index.html.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <title>React Form Builder</title>
  <link href="/Content/FormBuilder/font-awesome.min.css" rel="stylesheet" />
  <link href="/Content/FormBuilder/bootstrap.min.css" rel="stylesheet" />
  <link href="/nodeModules/react-form-builder2/dist/app.css" rel="stylesheet" />
</head>
<body>
  <script>
    var FORM_ACTION = "/testing";
    var FORM_METHOD = "POST";
  </script>

  <div class="clearfix" style="margin: 10px; width:70%">
    <h4 class="pull-left">Preview</h4>
    <button class="btn btn-primary pull-right" style="margin-right: 10px" id="button-
preview">Preview Form</button>
    <div class="modal" id="preview-dialog">
      <div class="modal-dialog">
        <div class="modal-content">
          <div class="modal-body">
            <div id="form-generator"></div>
          </div>
          <div class="modal-footer">
            <button type="button" class="btn btn-default" data-dismiss="modal"
id="button-close">Close</button>
          </div>
        </div>
      </div>
    </div>
  </div>
  <div id="form-builder"></div>
  <!-- Load React. -->
  <script src="https://unpkg.com/react@16/umd/react.development.js" crossorigin></script>
  <script src="https://unpkg.com/react-dom@16/umd/react-dom.development.js"
crossorigin></script>
  <!-- Load our React component. -->
  <script src="/nodeModules/react-form-builder2/dist/app.js"></script>
  <script src="/Scripts/FormBuilder/form-generator.js"></script>
  <script src="/Scripts/FormBuilder/form-builder.js"></script>
</body>
</html>
```

Figure C.1 – FormBuilder.cshtml